AN ISO-CONTOUR PLOTTING ROUTINE AS A TOOL
FOR MAXIMUM LIKELIHOOD ESTIMATION

RAFAEL C. ANDREU

June 1975 WP 795-75

AN ISO-CONTOUR PLOTTING ROUTINE AS A TOOL FOR MAXIMUM LIKELIHOOD ESTIMATION

RAFAEL C. ANDREU

June 1975 WP 795-75

This Working Paper is part of Mr. Andreu's doctoral program at the MIT Sloan School of Management.

1.- Introduction.

Doing maximum likelihood estimation when the likelihood function is so complicated that it becomes very difficult to deal with it analytically ia a problem which has to be solved numerically.

The process of estimating the parameters of an underlying lognormal distribution when sampling is considered to be without replacement and proportional to size constitutes one such case. This particular sampling process is being used as a model for inferring the distribution of yet undiscovered oil and gas deposits from a sample of discovered pools. Pool discovery (observing deposit sizes in order of discovery) "is more akin to sampling without replacement and proportional to size than to sampling values of independent, identically distributed random variables" (1). This feature had been ignored until recently.

In addition, as it is also pointed out in (1), the population from which the discovery process picks pool sizes can be considered as a sequence of N values of N mutually independent random variables identically distributed with a common lognormal density, generated by nature. There is empirical evidence showing that a variety of actual size distributions can be reasonably characterized as being lognormal.

The mathematical analysis of sucha a sampling process is contained in (1), where an asymtotic expansion of the likelihood function (valid for large N and fixed sample

size n) is developed, since working with the exact expresions turns out to be very difficult.

In turn, since this asymtotic expansion is analytically complicated, to understand its behavior we supplement the treatment of it in (1) with numerical analysis of particular cases. Its complexity may be better understood by looking at its expresion. The expansion for the sampling density of $\underline{Y}$, valid for large $p = N-n$ is:

$$I_{N,n}(\underline{Y}) \prod_{j=1}^{n} Y_j f(Y_j|\underline{\theta}) = \{\frac{\Gamma(p+n+1)}{\Gamma(p+1)} \prod_{j=1}^{n} \frac{Y_j f(Y_j|\underline{\theta})}{[pM_1+b_j]}\}$$

$$\times \{1+\tfrac{1}{2}pVg_2(pM_1,\underline{Y}) + \tfrac{1}{8}p^2V^2g_4(pM_1,\underline{Y})$$

$$+ p(\tfrac{1}{6}M_3-\tfrac{1}{2}M_1M_2+\tfrac{1}{3}M_1^3)g_3(pM_1,\underline{Y}) + 0(p^{-3})\}$$

where the functions $g_m(pM_1,\underline{Y})$ are given by

$$g_2(pM_1,\underline{Y}) = \sum_{j=1}^{n}(pM_1+b_j)^{-2} + [\sum_{j=1}^{n}(pM_1+b_j)^{-1}]^2,$$

$$g_3(pM_1,\underline{Y}) = [\sum_{j=1}^{n}(pM_1+b_j)^{-1}]^3 + 3[\sum_{j=1}^{n}(pM_1+b_j)^{-1}][\sum_{j=1}^{n}(pM_1+b_j)^{-2}]$$

$$+ 2\sum_{j=1}^{n}(pM_1+b_j)^{-3},$$

$$g_4(pM_1,\underline{Y}) = [\sum_{j=1}^{n}(pM_1+b_j)^{-1}]g_3(pM_1,\underline{Y}) \qquad 1\text{-}3$$

$$+ 3[\sum_{j=1}^{n}(pM_1+b_j)^{-1}]^2[\sum_{j=1}^{n}(pM_1+b_j)^{-2}]$$

$$+ 3[\sum_{j=1}^{n}(pM_1+b_j)^{-2}]^2 + 6\sum_{j=1}^{n}(pM_1+b_j)^{-4}$$

$$+ 6[\sum_{j=1}^{n}(pM_1+b_j)^{-1}][\sum_{j=1}^{n}(pM_1+b_j)^{-3}] .$$

and where $\underline{Y}$ is a vector of random observations, $f(Y\,|\,\underline{\theta})$ is the lognormal density with vector parameter $\underline{\theta} = (\mu,\sigma^2)$, M_1 and M_2 are the first two moments of such density, N = population size, n = sample size and $b_j = \sum_{i=1}^{j} Y_i$.

Our main goal in studying this expansion is to obtain approximate maximum likelihood estimates (MLE) for the parameters of the underlying lognormal density, μ and σ^2, and for the finite population size N. Namely, defining the likelihood function for μ, σ^2 and N given a data vector $\underline{Y}$ as $l(\mu,\sigma^2,N\,|\,\underline{Y})$, we wish to find a triple (μ_0,σ_0^2,N_0) of values (μ,σ^2,N) that maximize l.

ML estimators may be analytically obtained in the limit, as $N\to\infty$; hence a question of interest is how large should N be to actually obtain in practice a MLE of all three parameters μ, σ^2 and N. It is also interesting to get an idea of how stable these estimates turn out to be as a function of the sample size.

The approach we took to numerically analyze the behavior of the likelihood function for particular samples was to construct iso - contour graphs of it. Doing so in two dimensions with one parameter fixed allowed us to visually de-

tect interesting properties of the function.

What we did was the following: Taking the population size N as fixed and regarding μ and σ^2 as arguments of the likelihood function, we generated sets of points in the (μ, σ^2) plane satisfying

$$l\,(\mu, \sigma^2, N \mid \underline{Y}) = k,$$

with k fixed. (Such sets of points are called iso - contours). Plotting sets of such iso - contours for different values of the population size and different values of the sample size, we gained some insight into properties of the likelihood function.

In what follows we will describe these properties , along with the estimates we obtained for the parameters mentioned above.

In a different context, certain methodological details of the design of an iso - contour generating routine turn out to be interesting as well. While it is easy to find a set of points satisfying an iso - contour equation (over given intervals for the function's arguments, and provided we have tabulated function values for points belonging to those intervals), what constitutes a more difficult problem is to order them in such a manner that a plotting device can go from one point to the next actually drawing the iso - contour curve. Thus, we will also describe the method we used in writing the routine which generates ordered sets of points belonging to a given contour. Some details are particularly relevant when one is interested in the function's extreme

points; these will be emphasized in the corresponding section. Program details, including actual program listings, are included in Appendix A.

As for the routine implementation, we have it working in the TROLL system environment, taking advantage of certain plotting facilities already available in this system.

A couple of macros which make the routine invocation easier from TROLL are included in Appendix B, along with their operating procedures.

Finally, some data referring to program requirements (memory and execution times) are outlined in Appendix C.

2.- Iso - contour generating routine. Outline of the method employed.

As stated in the previous section, the main difficulty lies on the proper ordering of the set of points belonging to a given iso - contour. The procedure we followed, inspired by that described in (2), orders the sets of points as it generates them and solves other problems that sometimes arise, such as having two or more disjoint iso - contour branches in the intervals of interest for the function's arguments.

Although some published material was available describing programs also motivated by the method introduced in (2) ref. (3), (4) , they were badly documented and not particularly well suited to interface with TROLL. Further, some interesting problems arise when generating an iso - contour near the function's extreme points, which were precisely the ones we were more interested in. Our final procedure differs slightly from that in (2) regarding to these problems, and so we thought it was worthwhile to briefly describe it here.

A general outline of the method will be presented first, so that those problems may be more easily understood.

What the routine accepts as input data is a table of function values in the points of a rectangular grid defined by a set of rectangular intervals of the function's arguments, with increments that may vary along the intervals. In such a grid, two units turn out to be of importance when

the aim is to generate iso - contours; namely the grid edges and the grid cells.

Grid edges are used as units during the first step of the procedure, in which each one is checked to see whether or not it is crossed by the iso - contour being built. A way to conduct such a check is to test the condition

$$(f_1 - k) \cdot (f_2 - k) \leqslant 0 \qquad \text{(a)}$$

for every edge in the grid, where f_1 and f_2 are the function's values at the edge extreme points and k is the level of the desired iso - contour. Successful tests are recorded during this first step, so that when it is finished (having tested all the grid edges) , we are left with a set of grid edges crossed by the iso - contour under construction.

Two properties of this procedure must be emphasized at this point. First, test (a) will only detect at most one crossing point at any edge (so that when the edges' size is too big, situations such as the one depicted in Fig. 2.1 will not be properly recorded), and points of tangency

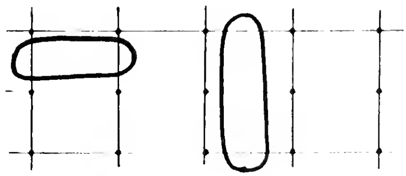


Fig. 2.1

between grid edges and iso - contour lines will not be recorded at all. Second, the procedure will go on in its second

step to locate the points of intersection between edges and contour lines, so that the points contained in the resulting iso - contour set will all lie on grid edges.

The second step of the procedure takes grid cells as working units, and relies on the fact that when the grid size is properly chosen, either none or two edges of a grid cell will be crossed by an iso - contour line (in general, if tangency points are not recorded, an even number of edges will be crossed in any grid cell; when the edges size is too big, again, a cell may be crossed in its four edges, this is a situation likely to come out near the function's extreme points and is one of the problems that will be analyzed below). What this second step does is to detect an iso - contour starting point (by locating any crossed edge resulting from step 1, and giving preference to boundary edges to properly locate possible incomplete iso - contour lines in the region defined by the grid, as these have a well defined starting point at the boundary), and follow the contour line through adjacent cells to the one containing the starting point. This operation of following up the line is an iterative one which is done as follows:

1.- Having located the edge containing the starting point, set it up as "current edge".

2.- Delete any record regarding the "current edge" as being crossed to avoid coming back to it thus building up endless loops.

3.- Locate an iso - contour point in the "current edge"

by means of some interpolation procedure between edge extreme points; save it as next contour point.

4.- If a "current cell" exists, go to point 5; otherwise decide on one grid cell to which the "current edge" belongs (in general, there will a choice among two, unless that edge is at the boundary; either one will do); call it the "current cell".

5.- Of the four cells adjacent to the "current cell", pick up the one having the "current edge" in common with it (notice that this cell will have either none or one crossed edge, as crossing information about the common edge was previously deleted); call it the new "current cell".

6.- Check the edges in the "current cell" for crossing information; if a crossed edge is detected, call it the new "current edge" and go to point 2. When no crossed edge is detected, the iso - contour is complete. Record this fact and go to point 1 to look for other possible iso - contour branches.

Such an iterative process will be better understood by following the sequence of sketches in Fig. 2.2, where (I) shows the actual iso - contour being built and successive ones refer to procedure steps (a x stands for a crossed edge, a dot for an iso - contour point in the resulting set of points; the shaded cell is the "current cell", the darkened edge the "current edge").

At this point, we are in a better position to discuss the kind of problems that are likely to arise in the neigh

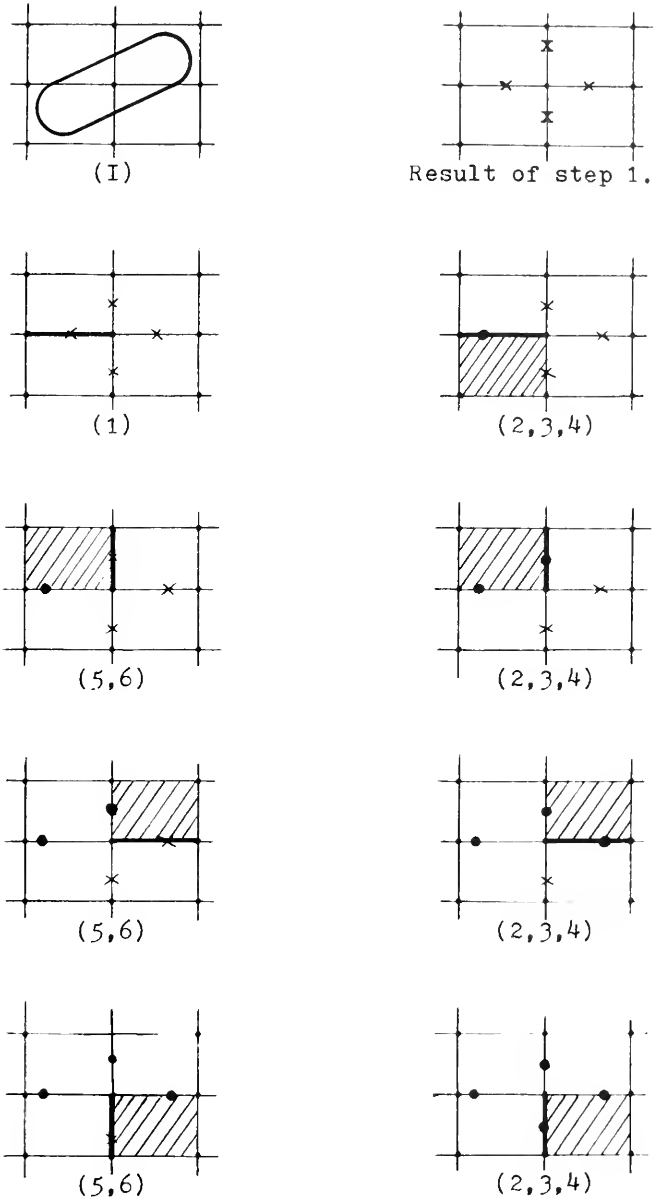


Fig. 2.2

borhood of function's extreme points. We will discuss three of them.

As has been emphasized repeatedly, grid cell size is critical; near extreme points this is even more true.

It may happen that, the grid size being too large, an small iso - contour near a function extreme point will be roughly contained in one grid cell, as depicted in Fig. 2.3.

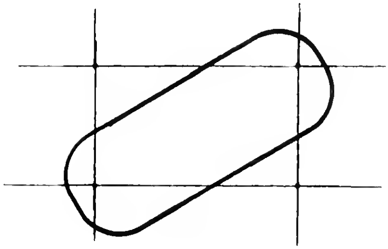


Fig. 2.3

Such a situation violates the assumption that either none or at most two edges were crossed on any given cell. There is no way of deciding in which way the four points in the cell should be joined to generate the iso - contour curve. There are three possible ways of joining them as shown in Figure 2.4. The way that eventually will bc used by the procedure

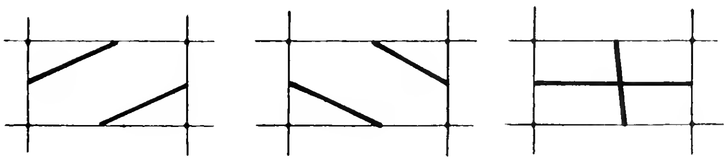


Fig. 2.4

described above depends on a variety of factors, such as on whether it picks one of the cell four crossed edges as star-

ting edge or not and the implicit sequence on which cell edges are checked for crossing information upon having it as current cell.

No reasonable way of solving this problem without changing the cell size (so that the four points will no longer belong to the same cell) was found, unless assumptions are made about topological properties of the function being analyzed. In our case, it is not safe to make any of these assumptions - at least in principle. In addition, given that one of the goals of our study is to determine extreme points, a finer grid is needed in any case to obtain precise estimates. Hence we decided to apply a finer grid whenever this problem appeared. Curiously, this is another situation which points out the weaknes of software procedures in dealing with pattern recognition; most of the early approaches to such problem were very similar ;i.e., superimposing a grid to the object under inspection and being unable to infer deatails from the general pattern. In our case, it is obvious that a human being, with the help of nearby iso - contours would have no problem in joining the four points in that cell in the correct way. The procedure doesn't generate such information, as it constructs one iso - contour at a time, but there is no apparent way to make (it) available to the routine to solve the problem unless very inefficient procedures, such as slope matching are used.

A second problem, particularly relevant when the function under study is very flat near the extreme points (and this was our case, as it will be seen later), is to get

a set of adjacent grid points all with the same function value, in turn equal to the level of the iso - contour being constructed. The situation is illustrated in Figure 2.5. Assume the values besides grid points are the function values at them, and that we are building the iso - contour of level 5. What

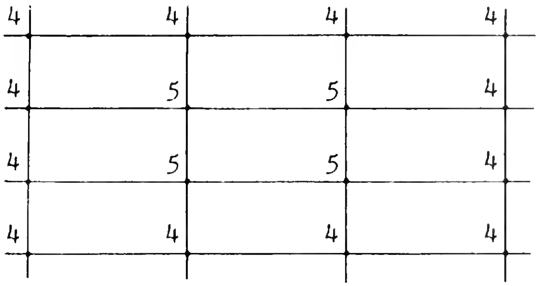


Fig. 2.5

seems logical to do is to record the edges with the maximum value at their extreme points as being crossed, since they themselves are so closed to the actual iso - contour. Test (a) above would indeed mark those edges as being crossed, but it would as well mark all the other edges with an extreme point of level 5. Then, what may happen (depending upon the same kind of factors outlined for the previous problem), is that we may end up with an iso - contour plot as ridiculous as the one depicted in Figure 2.6, consisting of four independent points considered as iso - contour disjoint branches.

Reference (2) proposed a way of avoiding this problem by disturbing the function values slightly in the proper direction so that only the outer edges would be recorded

as crossed (notice that when those points are at a relative minimum the disturbance has to be in the opposite direction).

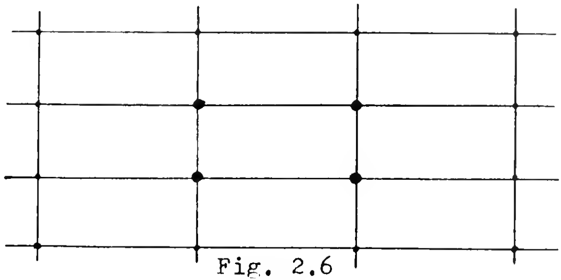


Fig. 2.6

This preserves the nature of test (a), but search for such grid nodes has to be undertaken as a pre-step for the method in use. The way we solved it is by complementing (a) in such a way that edges with extreme values both equal to the iso - contour level are not recorded; only those with a non zero difference at one of their extreme points are so. It is equivalent to the other one and it does not require any pre-step.

Finally, the problem depicted in Figure 2.1 is one which is a direct consequence of the cells being to large. It can not be solved by any other procedure than using a finer grid, since there is no way of detecting two crossing points in an edge with information about edge extreme points (and even if it were, the interpolation process would end up with the same points along the edges generating two exactly equal branches for the iso - contour.

3.- Asymptotic expansion behavior and maximum likelihood estimation.

We had at our disposal two sets of sample data to study the behavior of the likelihood function generated by them: the first is a set of 24 North Sea pools and the second is a set of 54 pools from a play in the Western Canadian Sedimentary Basin.

Before going any further, it is important to say that the chronological order in reservoir discoveries is of central importance, as the sampling process is assumed to be without replacement and proportional to -remaining- size.

Some of the results obtained using the expression of the likekliihood function on those samples are summarized in tables 1 and 2. In them, the values of μ and σ^2 which maximized the likelihood function for different values of N and sample size n are shown (When the sample size is less than the actual value stated above, it has to be interpreted that the sequence of first sample values was used as data, thus preserving chronological order). The estimates for N = ∞ were computed analytically, that is,

$$\sigma_\infty^2 = \frac{1}{n} \sum_{j=1}^{n} (\log Y_j - \log g)^2 \quad ,$$

$$\mu_\infty = \log g - \sigma_\infty^2 \quad ,\text{where}$$

$$\log g = \frac{1}{n} \sum_{j=1}^{n} \log Y_j \quad ,$$

and Y_j stands for the j^{th} observation.

TABLE 1

NORTH SEA DATA

n	N	μ_o	σ_o^2	Likelihood value	Figure #
24	200	2.47	1,45	-128.279	1
24	300	2.45	1.6	-128.124	2
24	303	2.422	1.569	-128.12	3
24	400	2.3	1.7	-128.179	4
24	500	2.32	1.6	-128.269	5
24	600	2.35	1.6	-128.369	6
24	750	2.5	1.3	-128.463	7
24	999	2.6	1.2	-128.314	8
24	2000	2.6	1.2	-128.894	9,10
24	5000	2.6	1.2	-127.551	11,12
24	10000	2.6	1.21	-127.425	13
24	20000	2.6	1.21	-127.362	14,15
24	10^5	2.59	1.22	-127.31	16,17
24	∞	2.50286	1.27988		
20	300	1.95	1.95	-103.827	18
20	400	1.75	2.15	-104.01	19
20	∞	2.334	1.31567		
12	200	0.	5.2	- 59.7028	20
12	300	0.	5.	- 60.0069	21
12	∞	2.1806	1.48137		

TABLE 2

WESTERN CANADIAN SEDIMENTARY BASIN DATA

n	N	μ_o	σ_o^2	Likelihood value	Figure #
52	500	2.2	2.3	-314.024	22
52	1000	1.85	2.5	-310.084	23
52	1500	1.7	2.6	-308.58	24
52	2000	1.52	2.7	-307.813	25
52		1.2648	2.72134		
40	1000	1.85	2.8	-245.863	26
40	1500	1.5	3.	-244.921	27
40	2000	1.3	3.2	-244.442	28
40	3000	1.	3.5	-244.012	29
40		1.40947	2.95797		
30	1000	0.8	3.9	-183.915	30
30	1500	0.1	4.7	-183.342	31
30		1.26892	2.95797		
20	1000	-4.	12.	-126.765	32
20	1500	0.	5.7	-129.761	33
20		2.02979	2.74941		

The column labeled "Figure #" indicates the iso - contour plot corresponding to each row. In these plots, the axis labeled "SIGMA" actually corresponds to σ^2 values.

We will center the discussion in the values shown in Table 1, as we have a wider range for them and the operating characteristics of the asymptotic expansion for the likelihood function which we detected seem to apply to the values on Table 2 as well.

The main feature to be pointed out is the degree of instability which shows up for μ and σ^2 estimates as N stays constant and the sample size is changed. More concretely, as the sample size n increases, ML estimates (keeping N fixed) for μ and σ^2 vary systematically: the estimate for μ increases while that for σ^2 goes dawn.

Furthermore, as n takes values below a certain level, such changes in μ and σ^2 estimates become more apparent, taking values very different from the ones obtained for larger values of n. This is the case for n = 12 in Table 1 (even for n = 15 with those data such a behavior begins to be very apparent) and for n = 30, 20 in Table 2.

There are, as we can see, two two main reasons for such instability.

The first is probably a consequence of sample data variability, which is relatively more important when the sample size is small. The characteristics of the samples we had -perhaps due to the fact that oil producing areas are so recognized when important discoveries take place, so that some early observations tend to be large- was such that

when used in the likelihood function produced a bimodal behavior which was hard to detect as the second relative maximum builds up in a region where the μ and σ^2 values are unreasonable as estimates of the underlying density.

Moroever, the second relative maximum tended to be more important as N increased for fixed n. As may be seen in the plots corresponding to N = 5000, 20000, 100000 when n = 24 for the first sample (Figures 11, 12, 14, 15, 16, 17), it reaches a point where becomes more important than the first one. When this happens, those values out of the reasonable ranges for μ and σ^2 mentioned above begin to appear.

For very small samples (e.g., 12 in Table 1, 20 or 30 in Table 2), it seems that the second relative maximum has taken over completely, and it is the only one obtained in those cases.

Before discussing the other reason for instability, more intrinsic to the nature of the problem and less related to data variability, it is perhaps worthwhile to make a methodological point for using the iso - contour routine or a similar tool in numerical analyses like the one described here. We would recomend a wide range on the function arguments to take the first pictures, forgetting about"reasonable" values around which the function is expected to behave in a certain way (e.g., to have a maximum). Having an overall idea of function behavior we can then concentrate on certain areas. If this step is ommitted, the first results tend to bias the analysis to pursue a certain region without paying any attention to others which may contribute heavily to explain peculiar be-

vior. In our environment, doing so implied a loss of detail, as the plots we generated were fixed in size, but even in this case it proved to be in the right direction.

The second cause of estimate instability was due to the extreme flatness of the function near its maximum(s). As it may be seen in the enclosed plots, this circumstance did show up rather strongly. Thus, small variations in the functional form due for example to small changes in sample data are likely to produce relatively important changes in the location of the function's extreme point(s).

The changes observed in the location of maximums were not, however, completely arbitrary. What happened was that the iso - contours we generated were roughly elliptic, and that the main axes of such ellipses tended to conserve their orientation in the μ - σ^2 plane. More concretely, the main axis tended to conserve its location, so that different estimates tended to lie in the line define by it. This may be seen as a common characteristic of all the plots we include.

As a function of N with n fixed, the iso - contour plots show that the function gets tighter along the small ellipses' axis, but being still very flat along the other one. This suggested that the underlying colinearity between μ and σ^2 estimates in the limit as $N \to \infty$ (see μ_∞ and σ^2_∞ equations above) could be somewhat preserved for finite values of N. Visual inspection of the plots pointed out a line equation close to

$$\mu = 3.775 - 0.8\,\sigma^2$$

for the first sample as main ellipses' axis. In the limit, for

n = 24, the equation is

$$\mu_{\infty} = 3.78274 - \sigma_{\infty}^{2} ,$$

not very far off the one visually fitted. This would explain the overall trend of μ and σ^2 estimates as the sample size n changes which we described at the begining of this section ; it may help to foresee their behavior for different sample sizes.

We were also interested in obtaining an estimate for N, the finite population size. Our results in this regard are not very conclusive. As shown in Table 1, consulting the column labelled "Likelihood values", it appeared as if for N = 303 we obtained an absolute maximum. However, as we had other results for increasing values of N, the function value increased above the quantity -128.12 corresponding to N = 303, and, what is worse, the overall maximum (up to -116.511 for N = 100,000 in Figure 16) is attained at a point with unreasonable values for μ and σ^2. Thus, at least for samples as small as the ones we had, ML estimators for N appeared to be quite instable, and other estimation methods should perhaps be tried to obtain a value for the population size. It has to be seen, however, how the asymptotic expansion works with regard to N for samples with a larger n. It may well be the case that our sample sizes were too close to values for which the second function's relative maximum begins to develop, thus jeopardizing the function usefulness in the relevant range for μ and σ^2.

To summarize, the two reasons for estimates' instability discussed above have pointed out how sample size

influences the estimates we obtain with the analyzed likelihood function expansion. Its pathological behavior for small samples seems to suggest that having more numerous samples would help to obtain better estimates, not only because they provide more information, but also because the likelihood function begins to behave better when n increases.

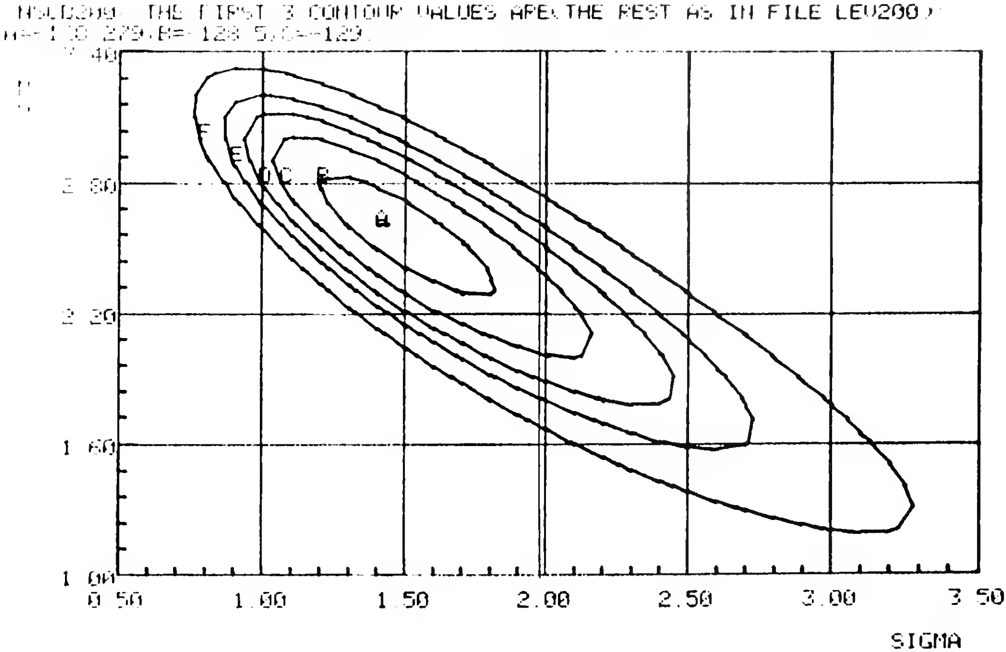

THE FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE LEV200)
A
B
C
D
E
F
2.80
2.20
1.60
1.00
0.50
1.00
1.50
2.00
2.50
3.00
3.50
SIGMA

Figure 1.

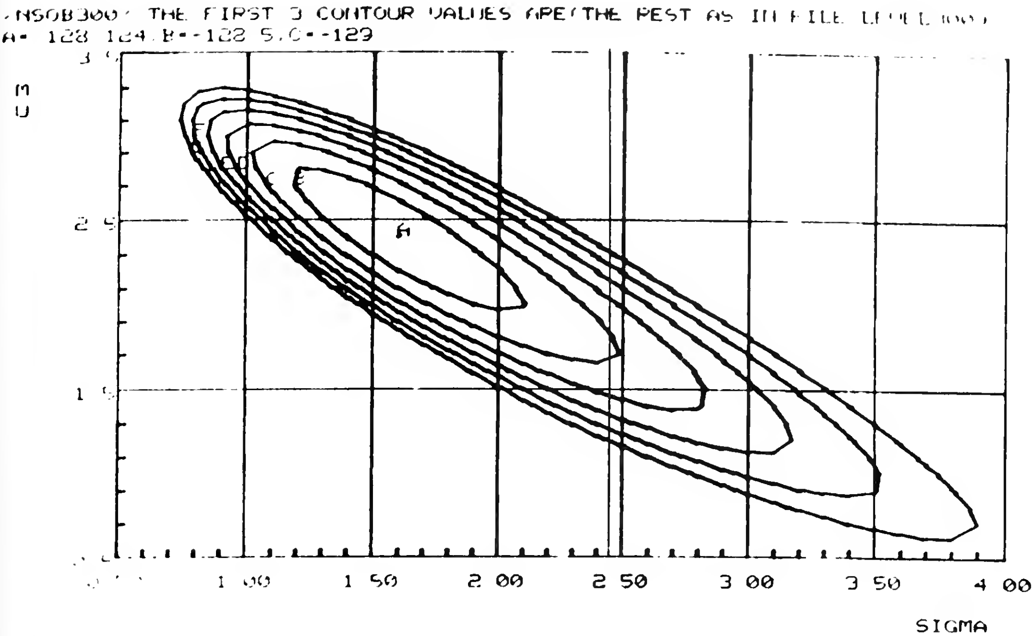
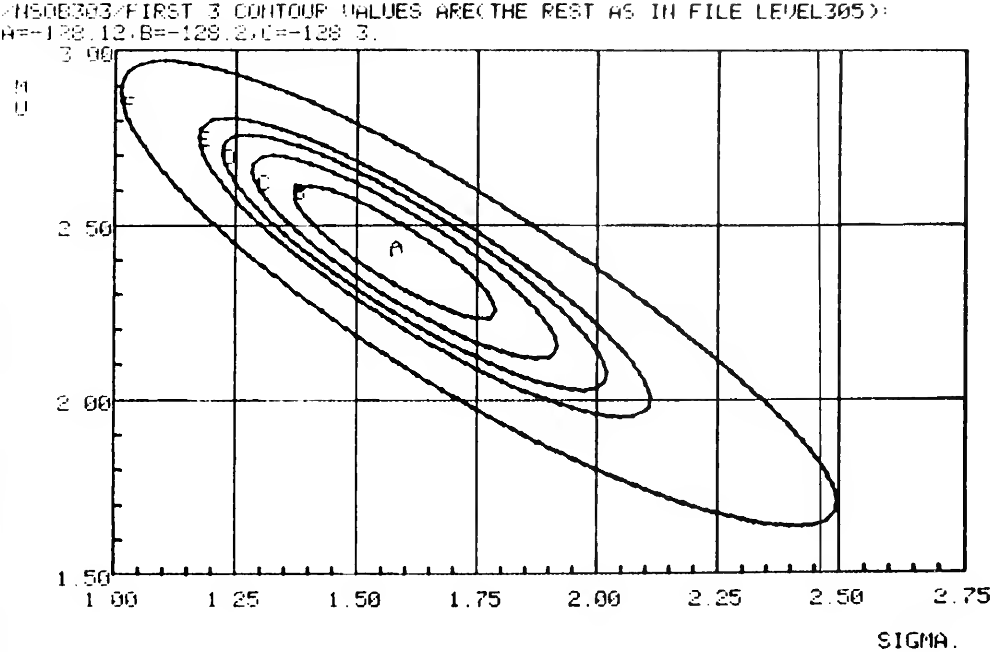
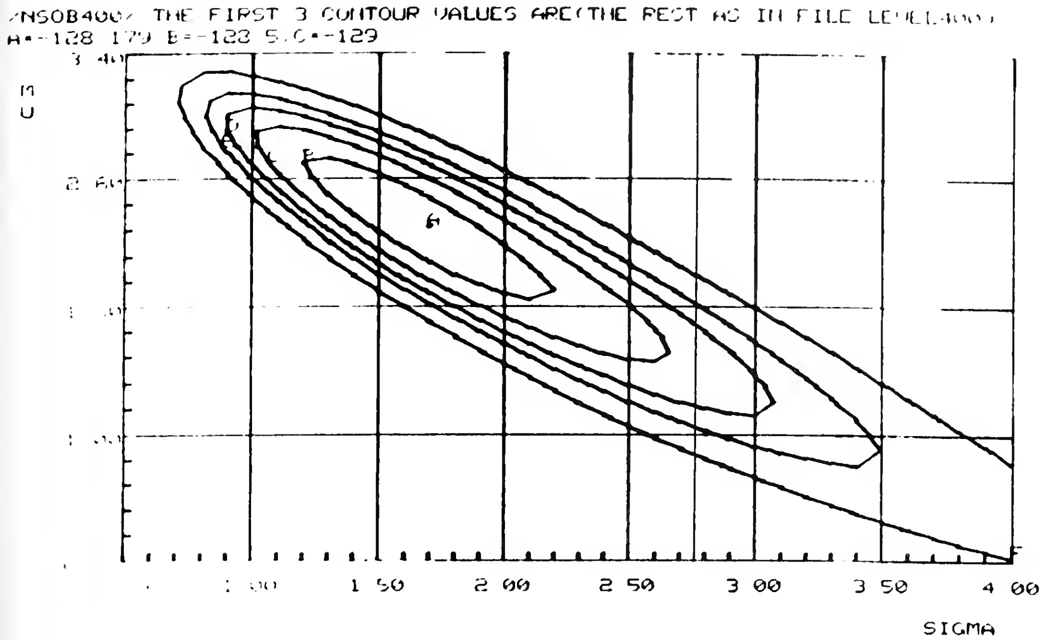

THE FIRST 3 CONTOUR VALUES ARE/THE REST AS IN FILE
M
U
1 00
1 50
2 00
2 50
3 00
3 50
4 00
SIGMA

Figure 2.



/NSDB303/FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE LEVEL305):
A=-128.12,B=-128.2,C=-128.3.
MU
SIGMA.
3.00
2.50
2.00
1.50
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
A
B
C
D
E
F

Figure 3.



/NSOB400/ THE FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE
A=-128 179 B=-123 5.C=-129
MU
1 50
2 00
2 50
3 00
3 50
4 00
SIGMA

Figure 4.

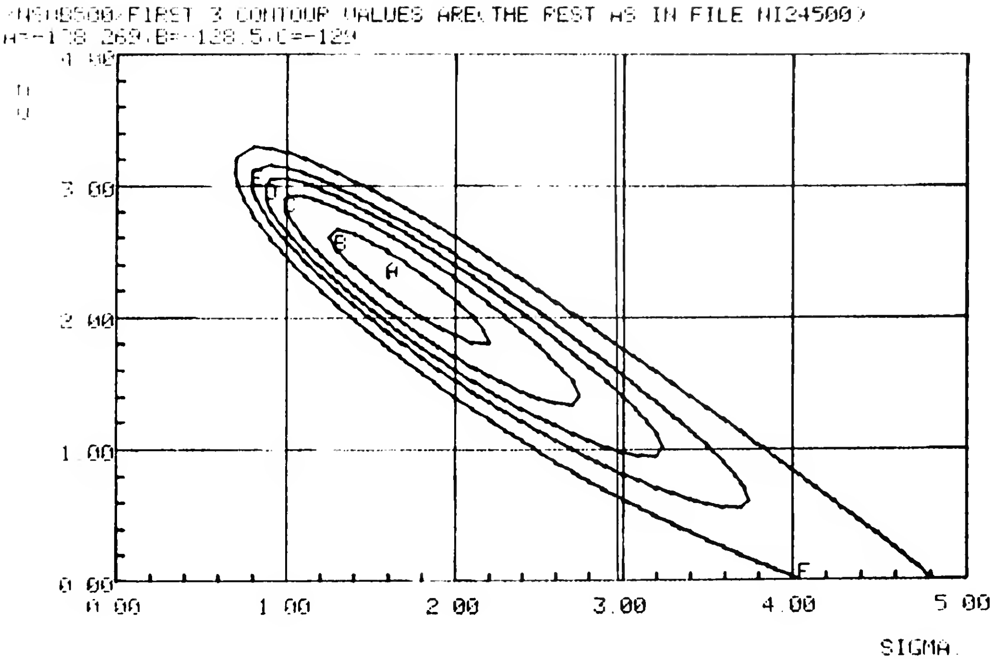
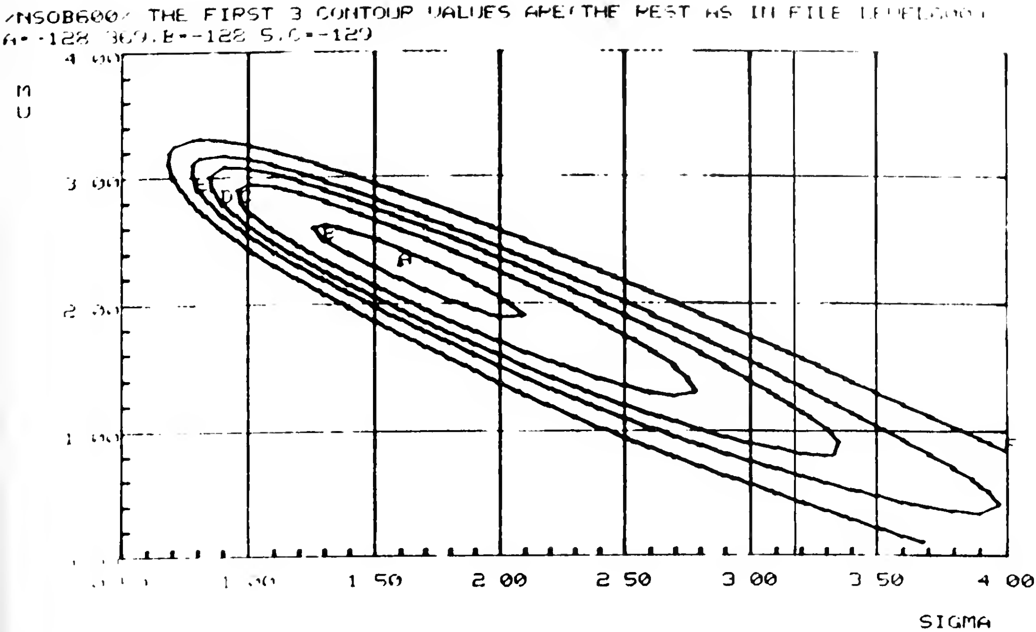

FIRST 3 CONTOUR VALUES ARE THE REST AS IN FILE NI24500
4.00
3.00
2.00
1.00
0.00
0.00
1.00
2.00
3.00
4.00
5.00
SIGMA.
A
B
C
D
E
F

Figure 5.



/NSOB600/ THE FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE
A=-128 369,B=-128 5,C=-129
M
U
SIGMA
1 50
2 00
2 50
3 00
3 50
4 00

Figure 6.

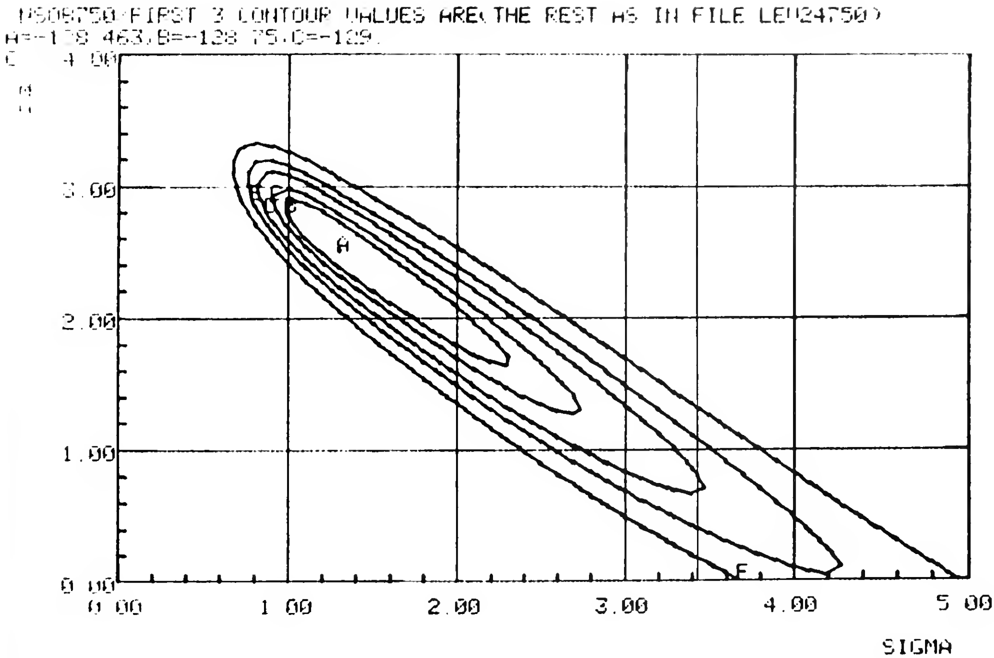

FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE
A
F
0.00
1.00
2.00
3.00
4.00
5.00
SIGMA

Figure 7.

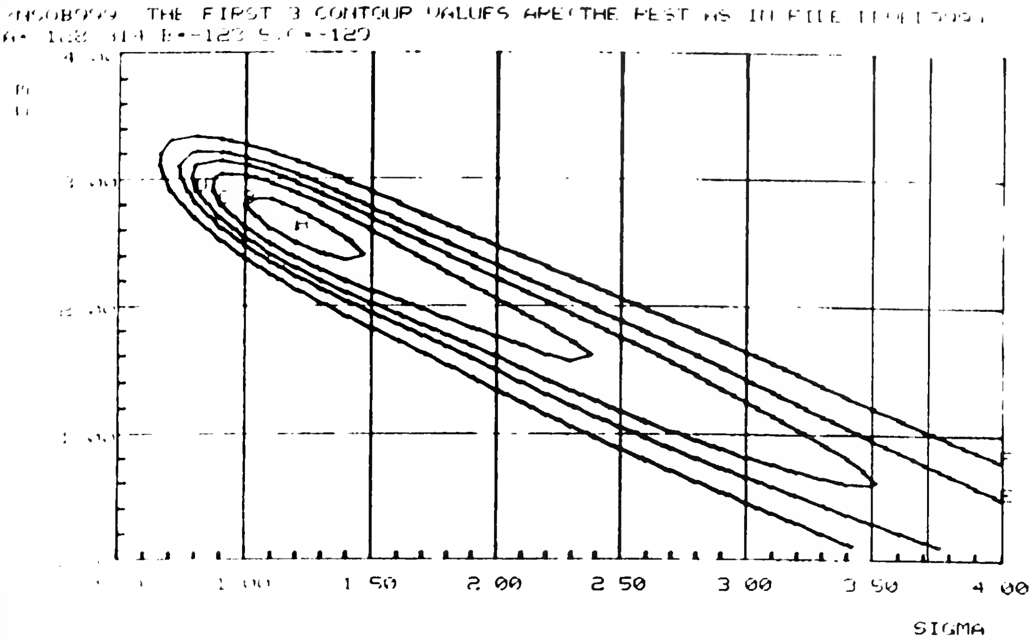


Figure 8.

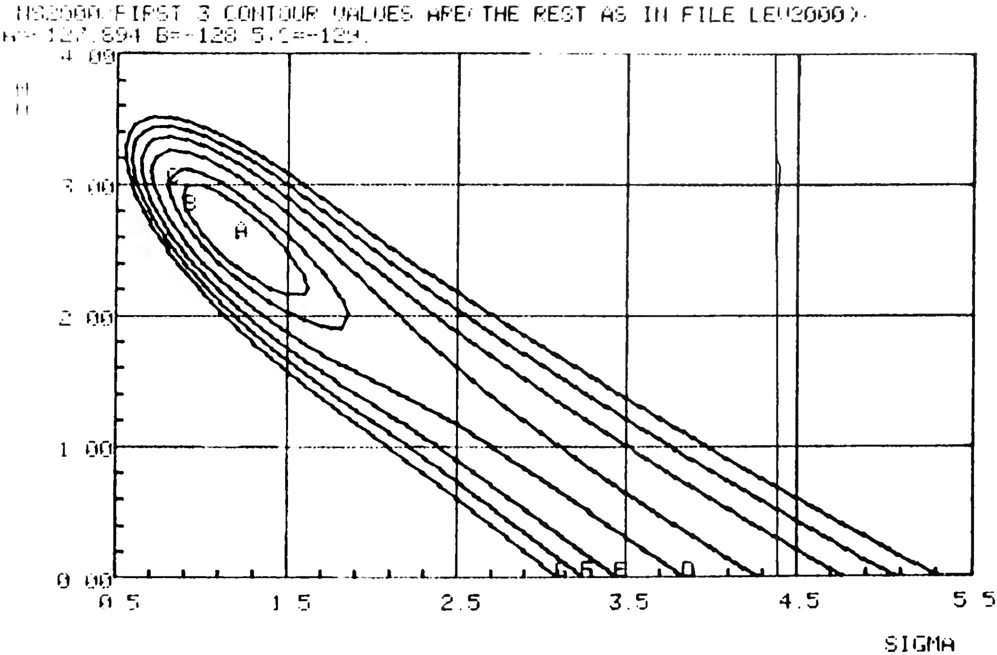

FIRST 3 CONTOUR VALUES ARE THE REST AS IN FILE
B=-128
4 00
3 00
2 00
1 00
0 00
A
B
C
D
E
F
G
0 5
1 5
2.5
3.5
4.5
5 5
SIGMA

Figure 9.

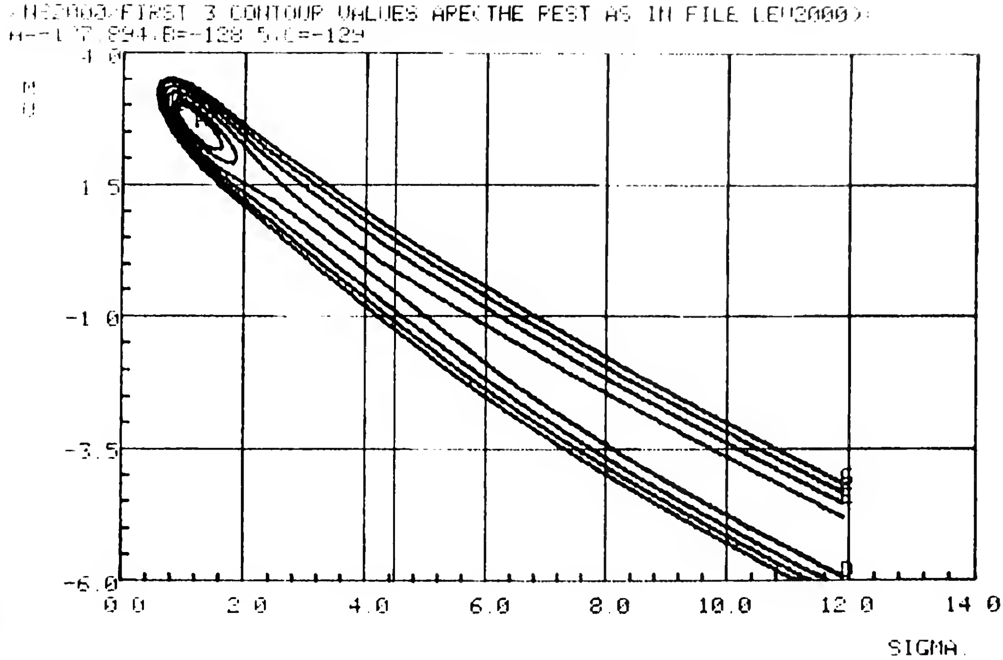
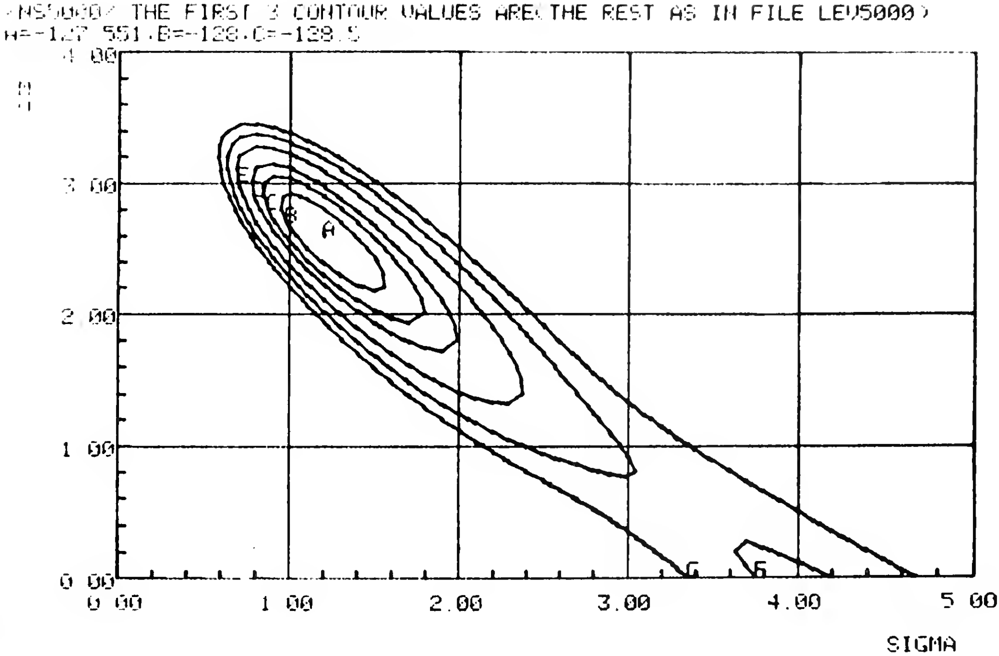

FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE
1.5
-1.0
-3.5
-6.0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
SIGMA

Figure 10.



THE FIRST 3 CONTOUR VALUES ARE<THE REST AS IN FILE LEV5000>
A=-127.551,B=-128,C=-128.5
4.00
3.00
2.00
1.00
0.00
0.00
1.00
2.00
3.00
4.00
5.00
SIGMA
A
E
G
G

Figure 11.

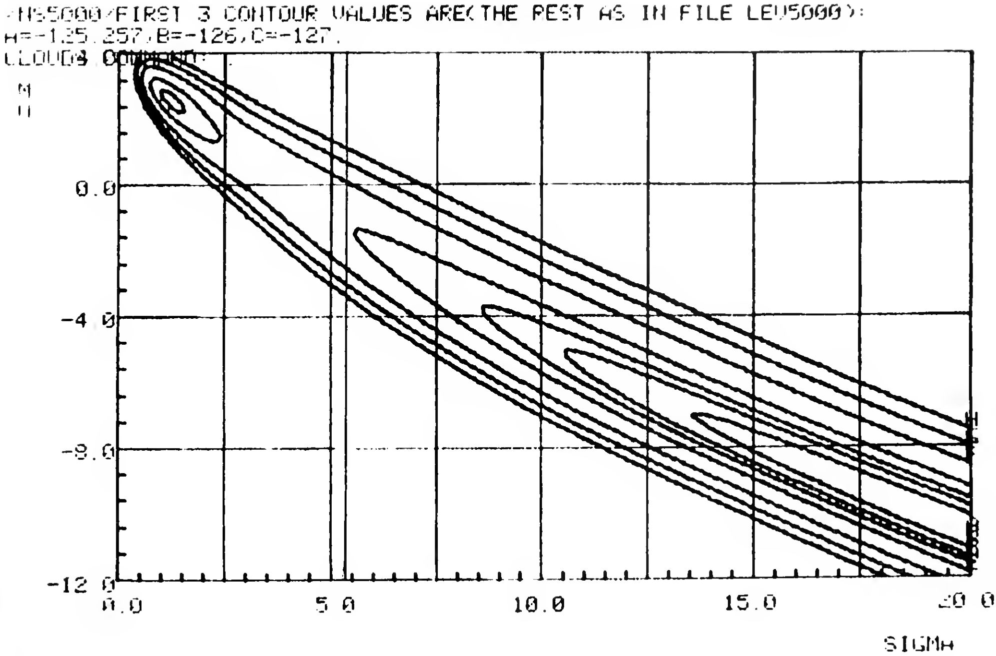
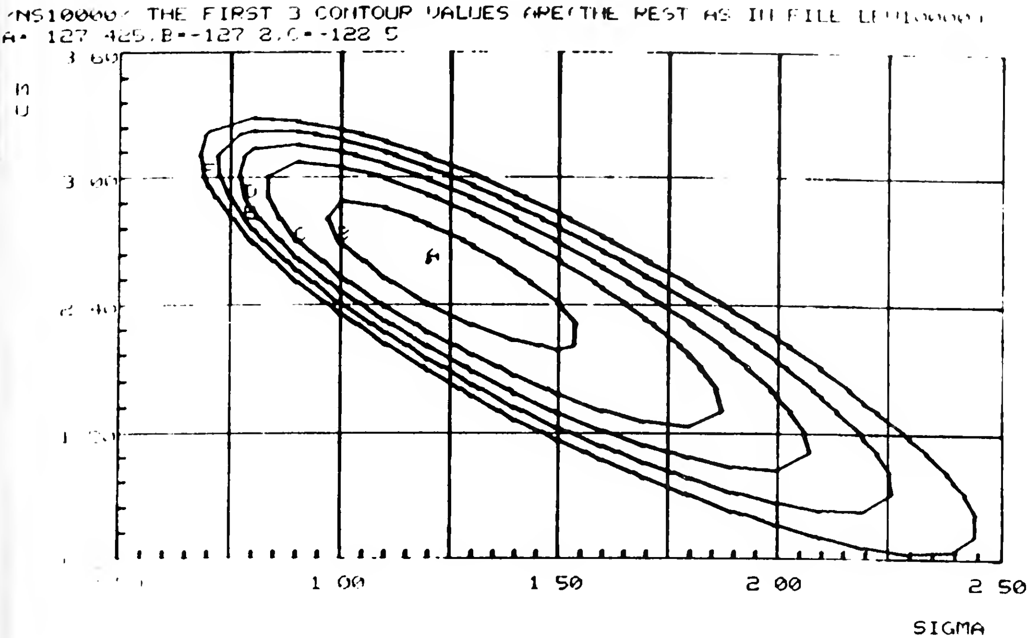

(HS5000/FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE LEV5000):
A=-125.257,B=-126,C=-127.
0.0
-4.0
-8.0
-12.0
0.0
5.0
10.0
15.0
20.0
SIGMA

Figure 12.



NS10000 THE FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE
A= 127 425,B=-127 2,C=-128 5
1 00
1 50
2 00
2 50
SIGMA

Figure 13.

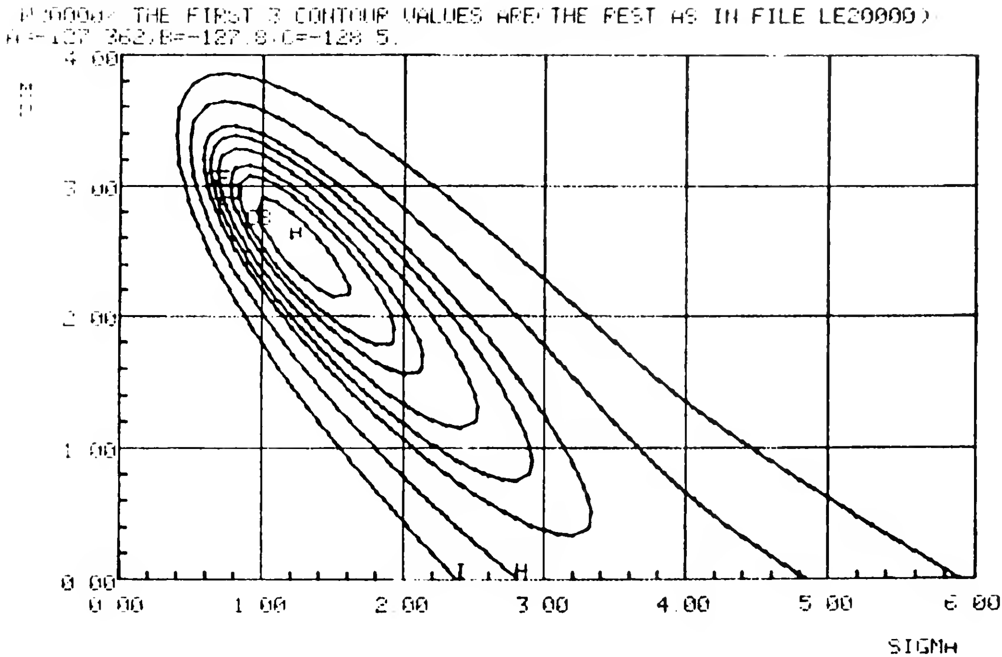

THE FIRST 3 CONTOUR VALUES ARE THE FEST AS IN FILE LE20000)
B=-127.8
C=-128 5.
4 00
3 00
2 00
1 00
0 00
0 00
1 00
2 00
3 00
4.00
5 00
6 00
SIGMA
H
I
H

Figure 14.

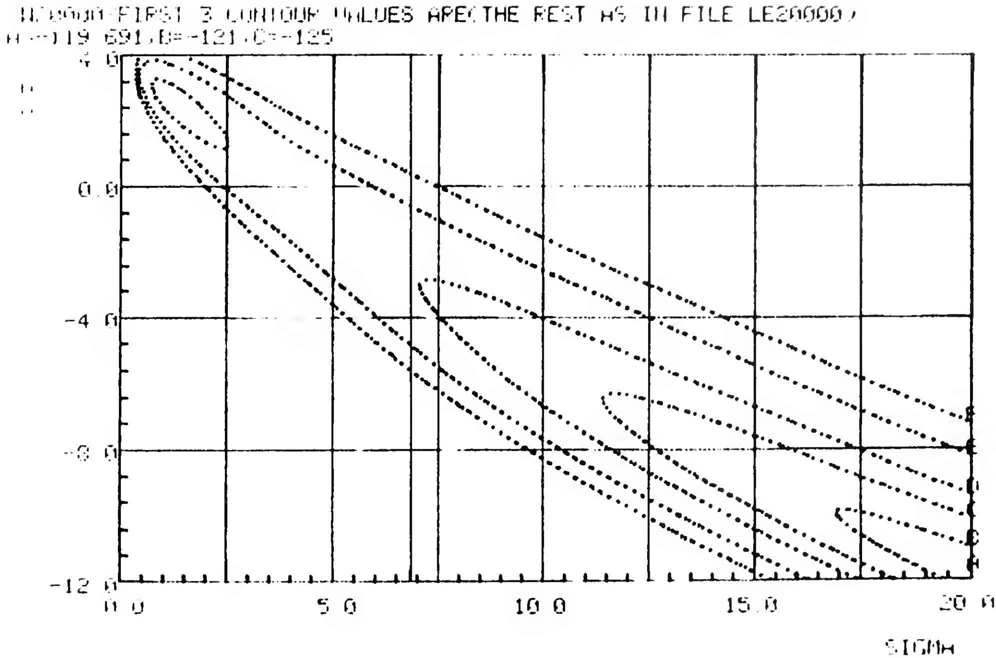
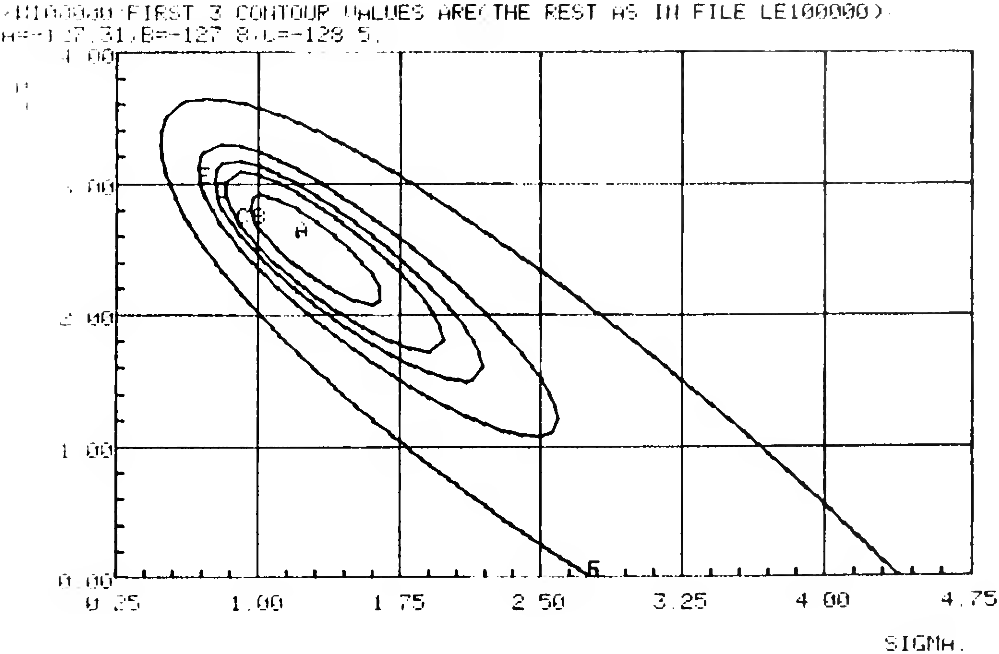


Figure 15.



FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE LE100000)
4.00
1.00
0.00
0.25
1.00
1.75
2.50
3.25
4.00
4.75
SIGMA.
A
B
C
E
6

Figure 16.

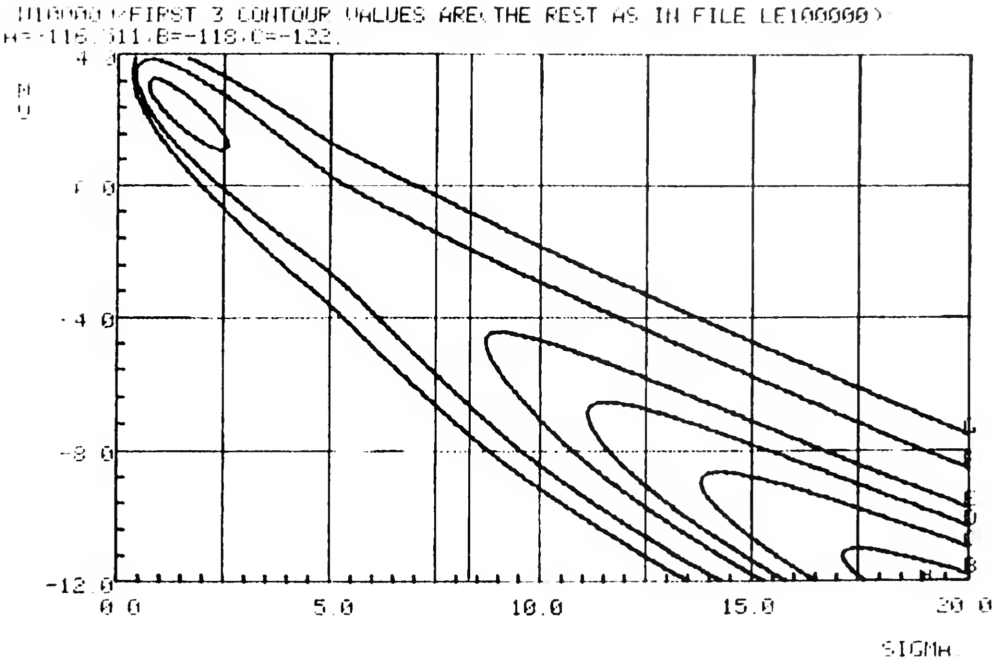
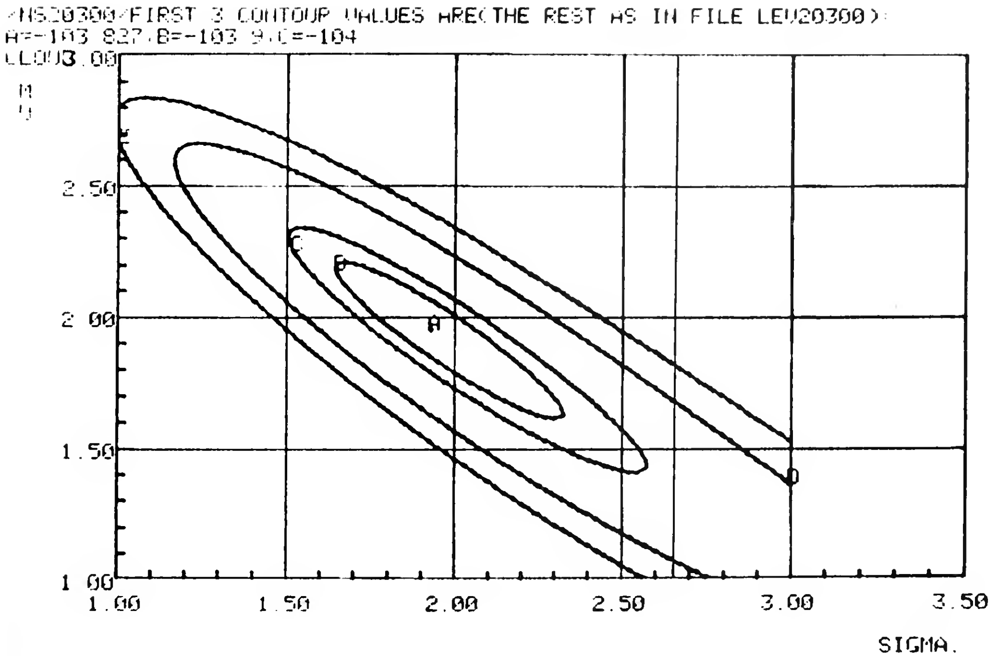
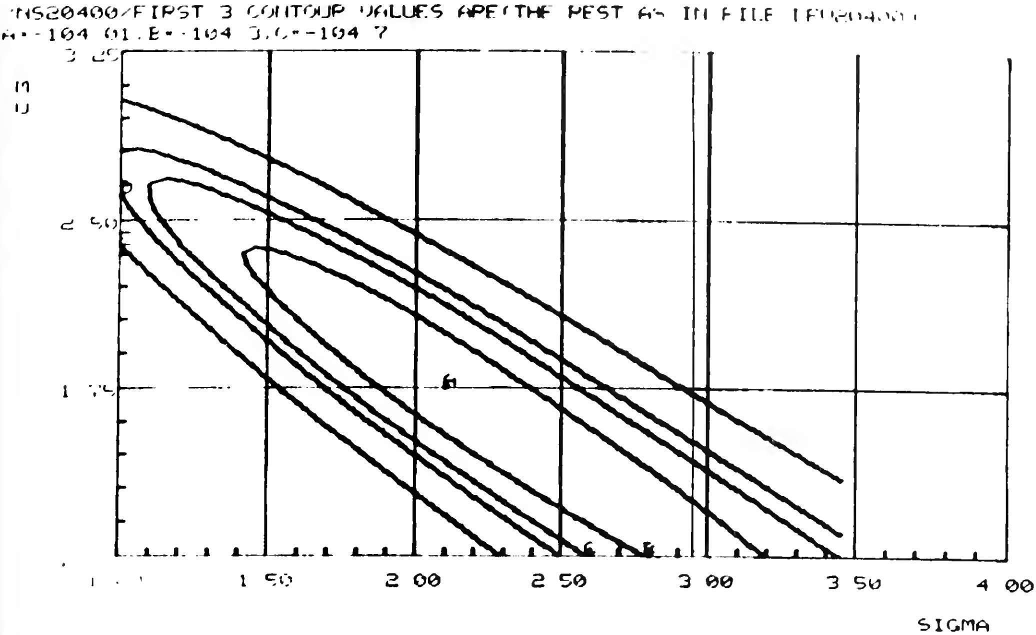

W100000 (FIRST 3 CONTOUR VALUES ARE, THE REST AS IN FILE LE100000)
A=-116.511, B=-118, C=-122.
MU
SIGMA
4.0
0.0
-4.0
-8.0
-12.0
0.0
5.0
10.0
15.0
20.0

Figure 17.



<NS20300/FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE LEV20300):
A=-103.827,B=-103.9,C=-104
LLOW
3.00
2.50
2.00
1.50
1.00
1.00
1.50
2.00
2.50
3.00
3.50
SIGMA.
A
B
C
D

Figure 18.



NS20400/FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE
1 50
2 00
2 50
3 00
3 50
4 00
SIGMA

Figure 19.

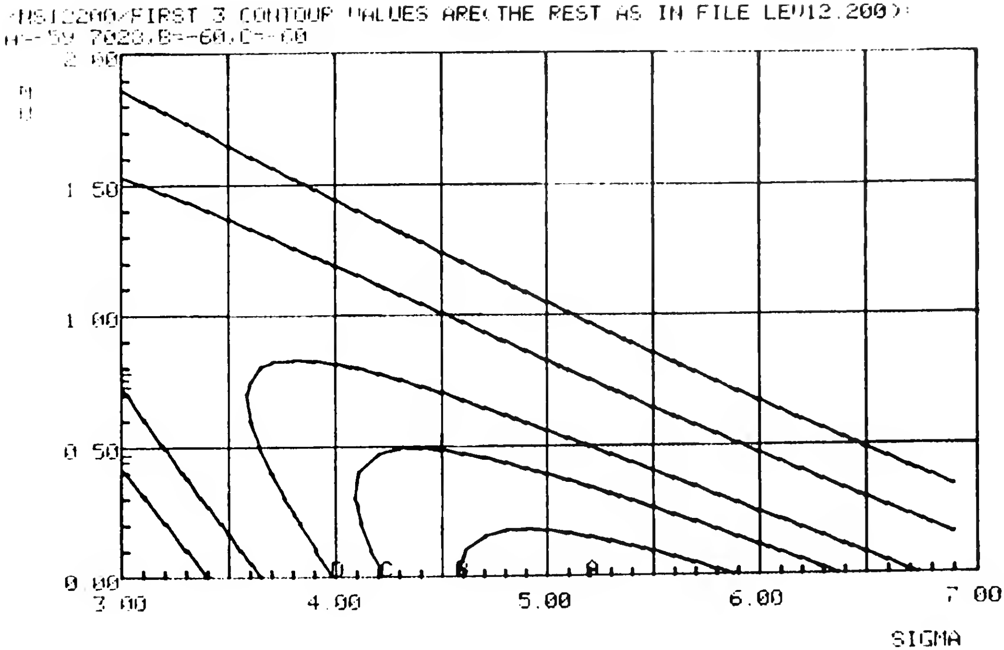

(NS12200/FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE LEV12.200):
A=-59.7020,B=-60,C=-60
2.00
1.50
1.00
0.50
0.00
3.00
4.00
5.00
6.00
7.00
SIGMA
D
C
B
A

Figure 20.

(NS12300/FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE LE(M12.300)
A=-60.0069,B=-60.5,C=-61.

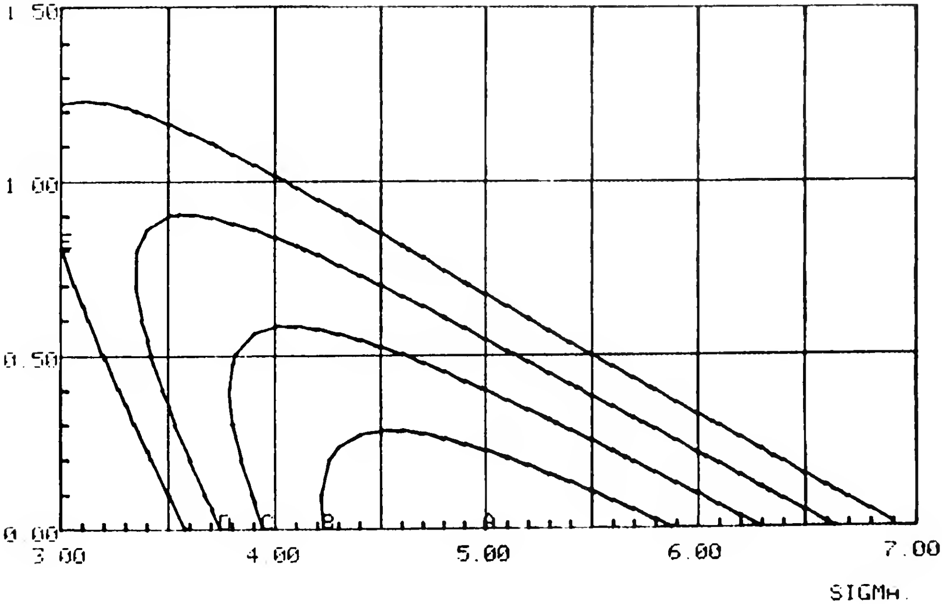
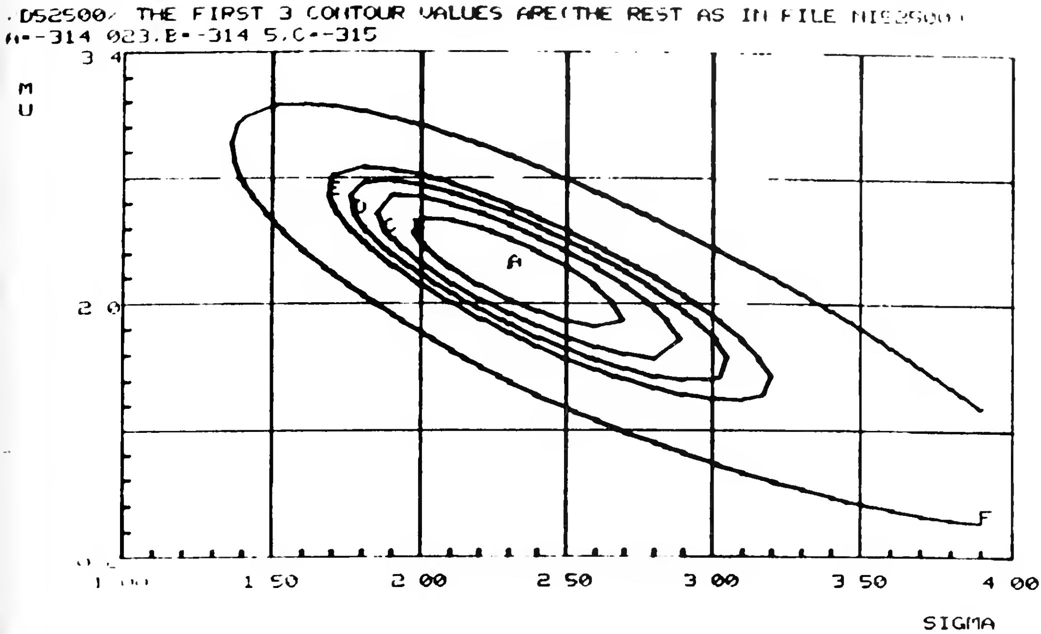


Figure 21.



.D52500/ THE FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE
A=-314 023.B=-314 5.C=-315
3 4
MU
2 0
1 50
2 00
2 50
3 00
3 50
4 00
SIGMA
A
F

Figure 22.

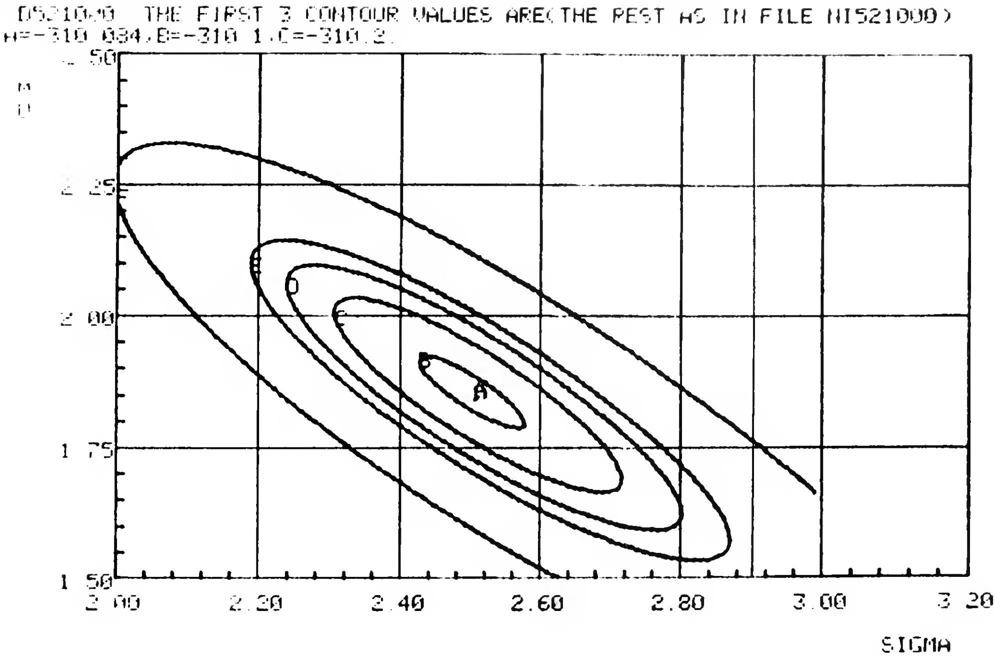

DS21000 THE FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE NI521000)
A=-310.034,B=-310.1,C=-310.2.
SIGMA
2.20
2.40
2.60
2.80
3.00
3.20
2.25
2.00
1.75
1.50
A
B
C
D
E

Figure 23.

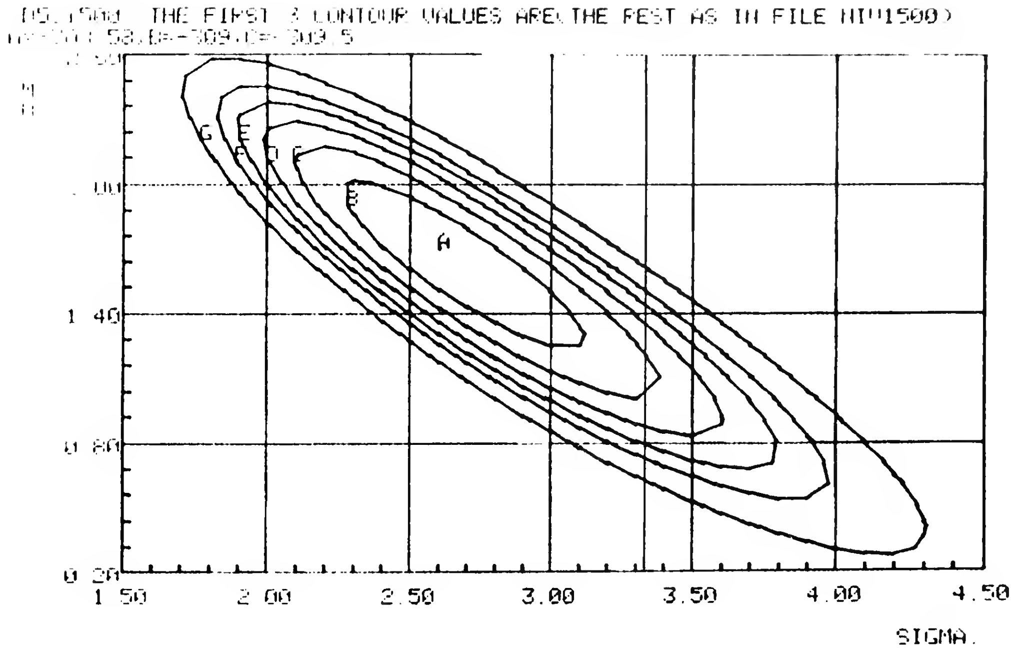

THE FIRST 3 CONTOUR VALUES ARE THE REST AS IN FILE
1.40
0.80
1.50
2.00
2.50
3.00
3.50
4.00
4.50
SIGMA.
A
B
C
D
E
F
G

Figure 24.

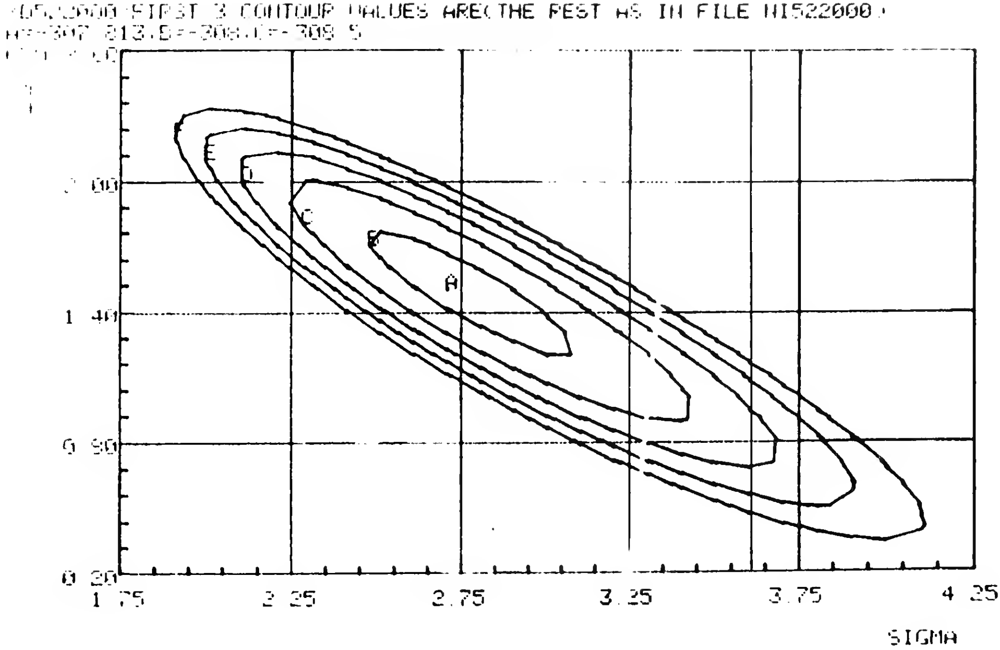

FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE HI522000)
2.00
1.40
0.80
0.20
1.75
2.25
2.75
3.25
3.75
4.25
SIGMA
A
B
C
D
E
F

Figure 25.

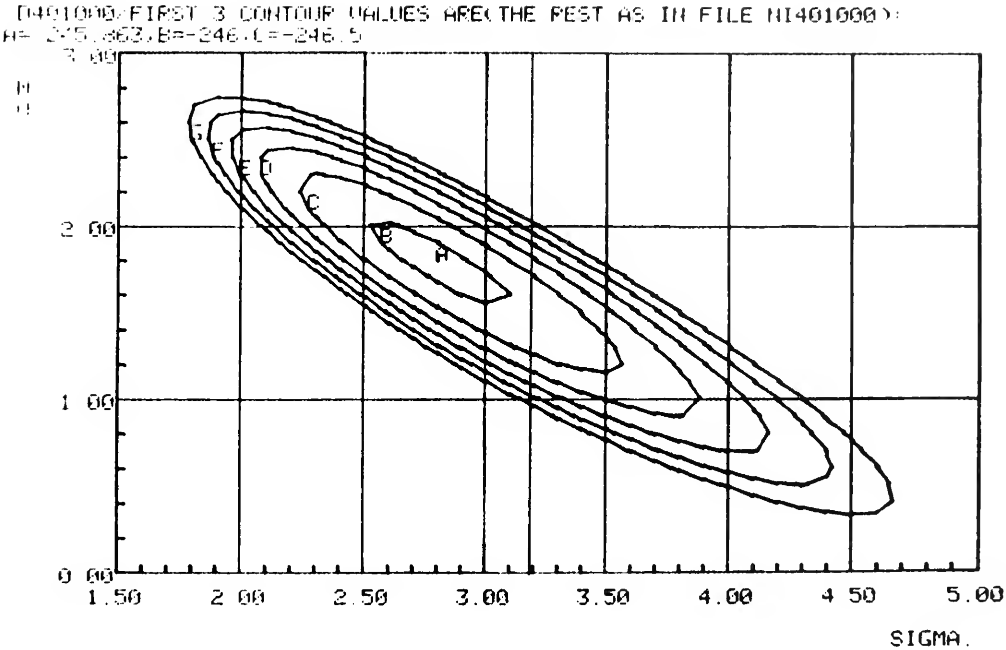

FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE NI401000):
B=-246,C=-246.5
2.00
1.00
0.00
1.50
2.00
2.50
3.00
3.50
4.00
4.50
5.00
SIGMA.
A
B
C
D
E
F
G

Figure 26.

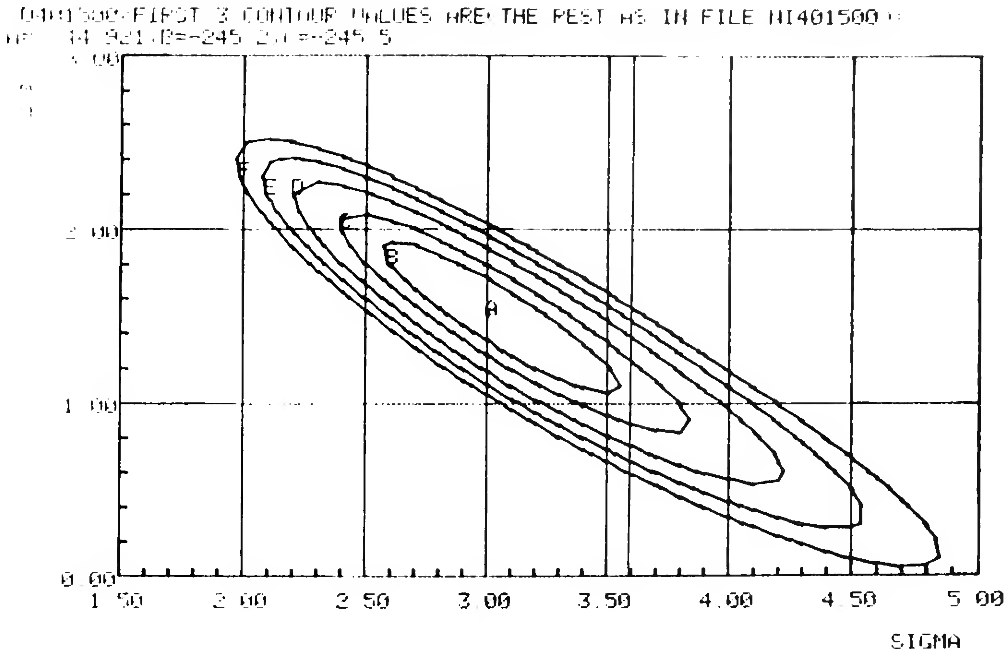

FIRST 3 CONTOUR VALUES ARE THE REST AS IN FILE NI401500
2.00
1.00
0.00
1.50
2.00
2.50
3.00
3.50
4.00
4.50
5.00
SIGMA
A
B
C
D
E
F

Figure 27.

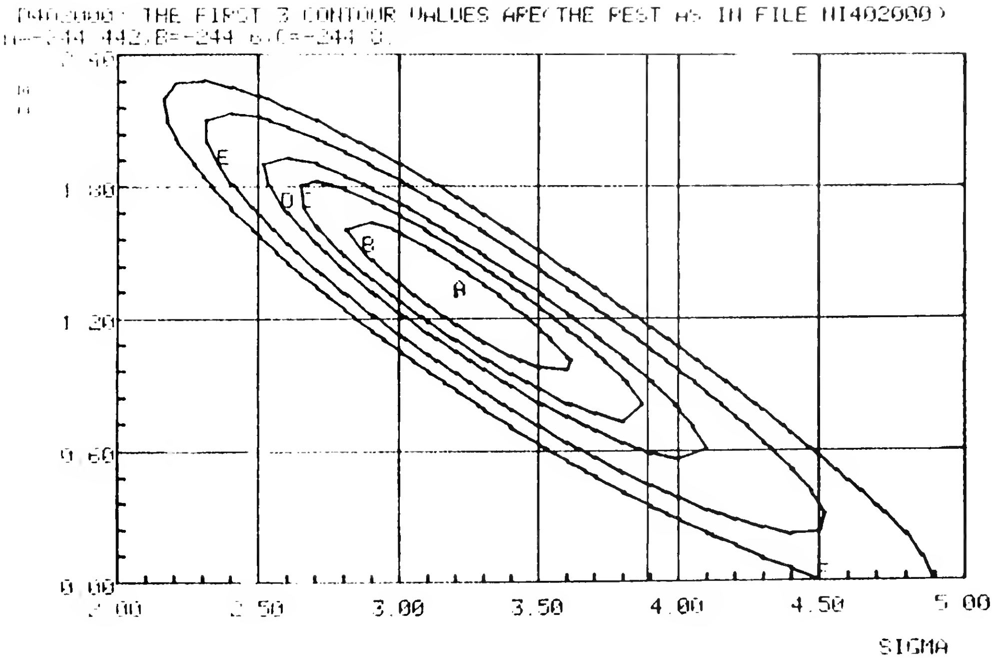

THE FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE HI402000)
1.80
1.20
0.60
0.00
3.00
3.50
4.00
4.50
5.00
SIGMA
E
D
C
B
A

Figure 28.

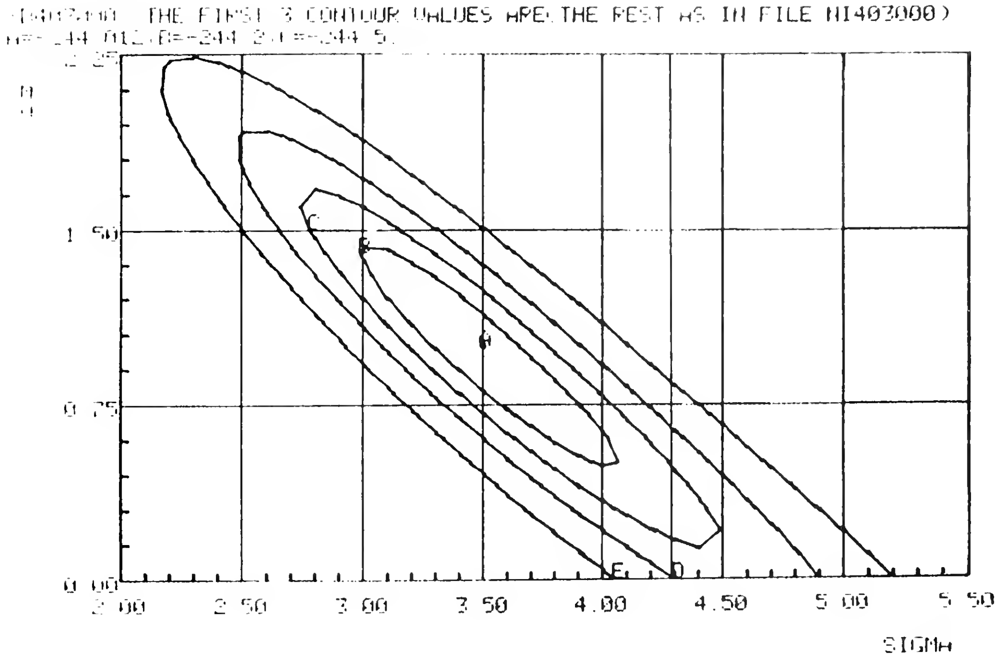

THE FIRST 3 CONTOUR VALUES ARE THE REST AS IN FILE HI403000)
1.50
0.75
0.00
2.00
2.50
3.00
3.50
4.00
4.50
5.00
5.50
SIGMA
A
B
C
D
E

Figure 29.

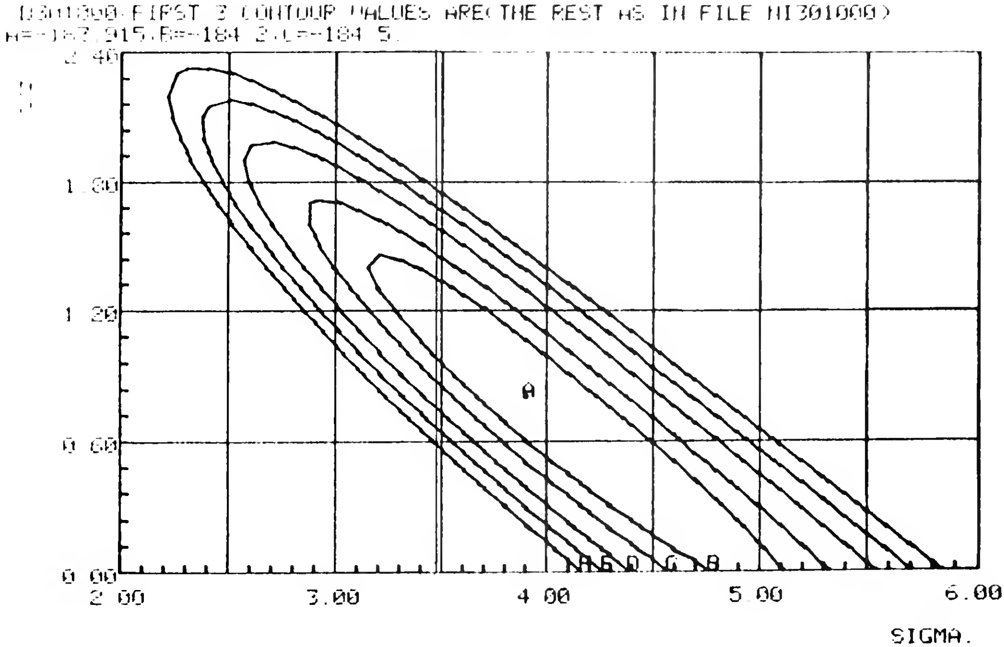

FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE HI301000)
915, B=-184 2, C=-184 5.
2.40
1.80
1.20
0.60
0.00
2.00
3.00
4.00
5.00
6.00
SIGMA.
A
A B D C B

Figure 30.

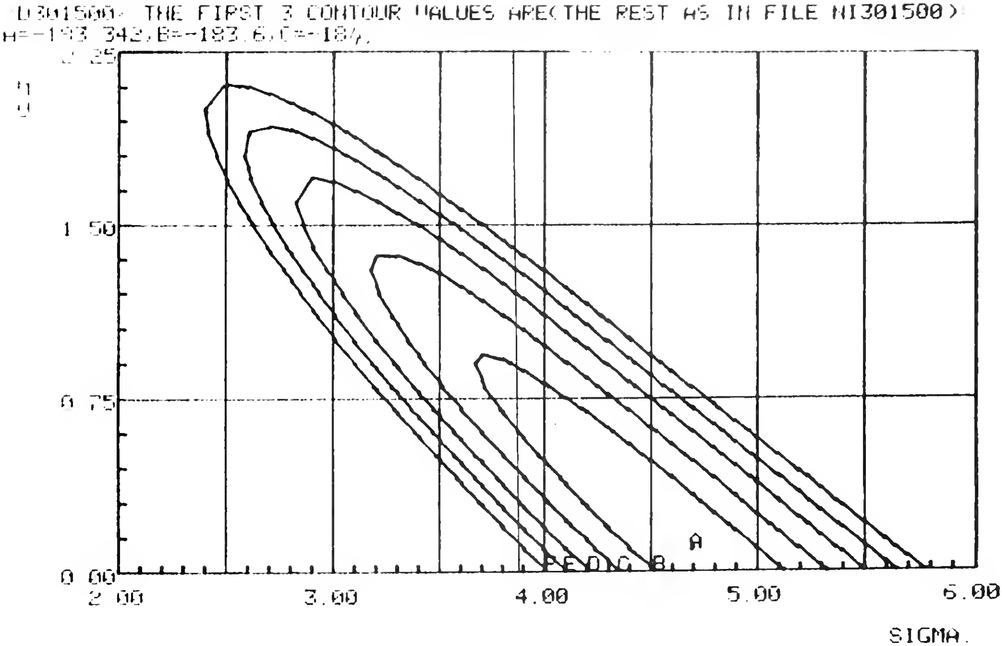

THE FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE NI301500)
B=-183.6
1.50
0.75
0.00
2.00
3.00
4.00
5.00
6.00
SIGMA.
A
F E D C B

Figure 31.

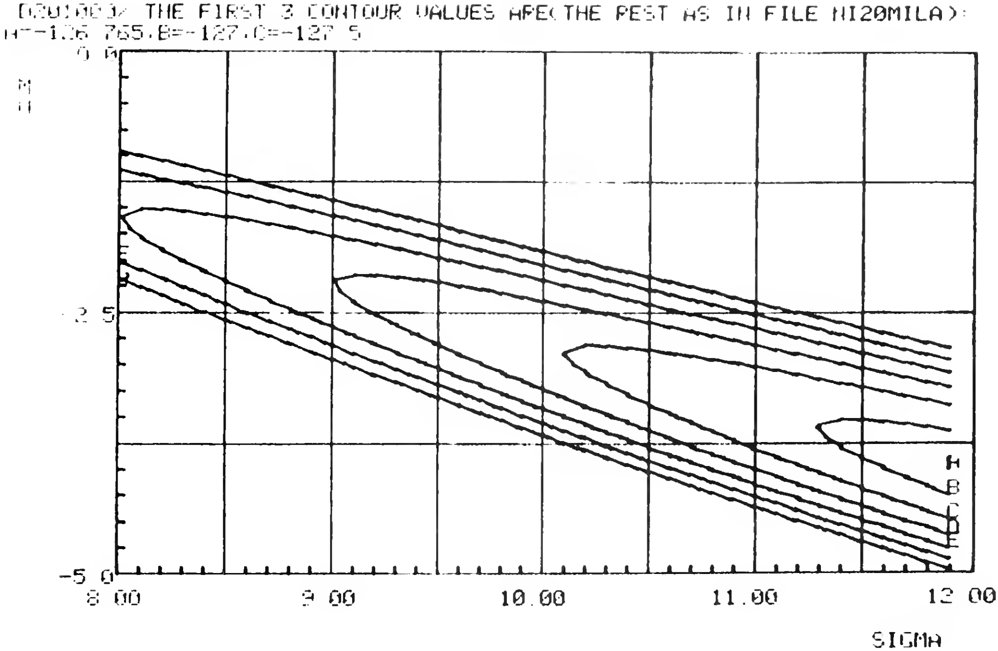

THE FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE NI20MILA):
A=-126 765,B=-127,C=-127 5
0 0
-2 5
-5 0
8 00
9 00
10.00
11.00
12 00
SIGMA
A
B
C
D
E

Figure 32.

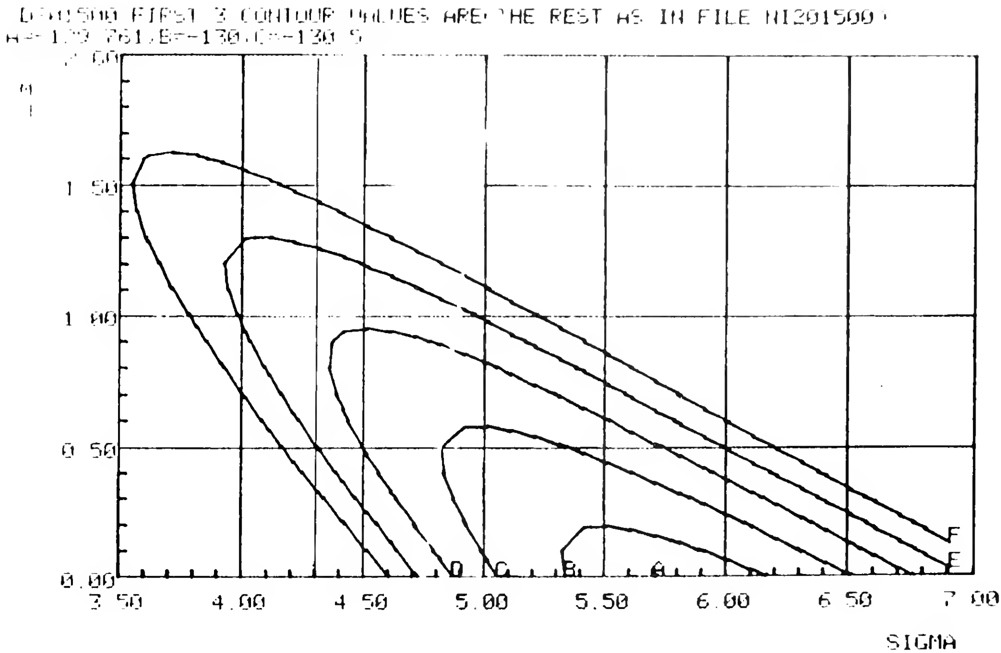


Figure 33.

APPENDIX A

Iso-contour plotting routine description and program listings.

The iso-contour generating routine consists of a set of FORTRAN and 370 Assembler subroutines which form a package. CTOUR is the name of the main routine in this package, and its purpose may be generally described as follows:

Given a tabulated function of two independent variables f(x,y), it determines sets of points belonging to the locus defined by the equation f(x,y) = C, where C is a given constant. (Subsequent calls to CTOUR with different C values allow to obtain a set of such loci).

1.- Procedure.

CTOUR employs as main procedure the one described in (1). Roughly, it works from a table of function values at points of a rectangular grid defined by two auxiliar input arrays (which permit the use of variable increments in x and y along their respective axes). Given a value for C in f(x,y)=C, CTOUR begins by marking all the mesh edges crossed by the locus defined by the above equation (an auxiliar bit memory is used to keep track of edges); then it computes the coordinates of the crossing points by linear interpolation between function values ad adjacent grid points. Points belonging to the locus are returned as result, ordered in such a way that it is possible to actually draw the contour by just joining them

(1) Automatic Contour Map. G. Cottafava and G. Le Moli, Comm. of the ACM, July, 1969.

as they are found in the result array. This result array may be logically divided in subtours, when the locus $f(x,y) = C$ is such that it posseses disjoint branches in the domain defined by the grid.

2.- Input and output variables description.

It is assumed that CTOUR will be called from a FORTRAN program. Its calling sequence is as follows:

```
CALL CTOUR(A,NR,NC,XS,YS,ALEVL,IA,PTS,NDPTS,NPTS),
```

where:

- A is a bidimensional array containing the tabulated function values. By convention, X values are constant along A's columns, while Y values are constant along A's rows. For convenience, A must be bordered by two extra rows and two extra columns which do not need to contain function values, but which <u>must</u> be included in the count to determine A's dimension.
- NR is the number of rows in A, as declared in the calling program (including the extra first and last rows).
- NC is the number of columns in A as declared in the calling program, also including the 2 extra columns.
- XS is an unidimensional array with NC elements containing the X values corresponding to different columns of A.
- YS is another unidimensional array with dimension NR containing the Y values corresponding to different rows of A. (2)

---

(2) In other words, $A(I,J) = f\,[XS(J),YS(I)]$.

-ALEVL is the function value for which the contour points are to be determined.

- IA is an auxiliary memory taking the form of an unidimensional integer array. It must contain at least $\lceil((NR-1)*NC+(NC-1)*NR)/32\rceil$ elements, and it must be declared accordingly in the calling program.

- PTS is a bidimensional array where the locus' points are to be returned. It is of the form PTS(NDPTS,2), where PTS(.,1) will contain X coordinates and PTS(.,2) Y coordinates for the resulting locus' points.

-NDPTS is the maximum number of points that may be allocated in PTS (i.e., its first dimension as declared in the calling program). If CTOUR reaches this maximum during the process of filling PTS, it calls another subroutine (EXTEND) which is supposed to save the contents of PTS when enough space to do so is available, cancelling the whole process otherwise. An actual program for EXTEND is not included below because it is likely to vary in different environments.

- NPTS is a result integer variable indicating how many points are returned in PTS.

3.- Example.

In what follows we describe how to set up the input variables for a call to CTOUR in a concrete case, along with the output generated by it.

Imagine that we have a function f(x,y) tabulated in the interval $[1 \leq x \leq 3,\ 2 \leq y \leq 5]$,by increments of 0.5 in both

x and y, and that we want to call CTOUR to obtain the set(s) of points belonging to the locus of level 4.5, that is, belonging to the curve defined by $f(x,y) = 4.5$.

If that is the case, the input data should be organized as depicted in Fig. A.1, and the grid being used would be the one shown in Fig. A.2.

Assume further that the iso-contour $f(x,y) = 4.5$ has the two branches drawn in Fig. A.2. Then, the result variables returned by CTOUR upon call with the Fig. A.1 variables would be of the form depicted in Fig. A.3. Notice that the first point of a contour branch is repeated as its last one to actually close it, unless the branch is incomplete with starting and ending points at the grid boundaries. Different branches are separated by delimiters labelled NA in both the x and y coordinates.

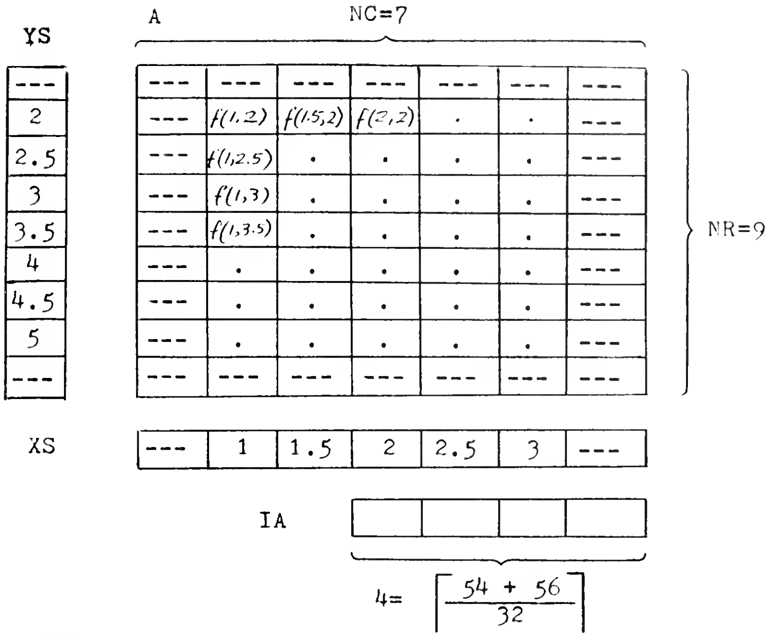
4.- Brief process description, with references to actual programs.

All subroutines are written in FORTRAN, with the exception of MARK, ERASE and SEE which manipulate bits in the auxiliary memory IA and are written in 370 Assembler for improved efficiency.

Fig. A.4 shows a high level flowchart for CTOUR, indicating calls to other subroutines in the package. It is quite straight forward, although some steps need further explanation. Following are succint discussions of such steps and the involved subroutines.

- "Clear auxiliary memory" just sets all the integer elements in IA to zero.

IA is used as a bit array, each bit corresponding to one



ALEVL = 4.5

Fig. A.1.- Input variables set up for example.

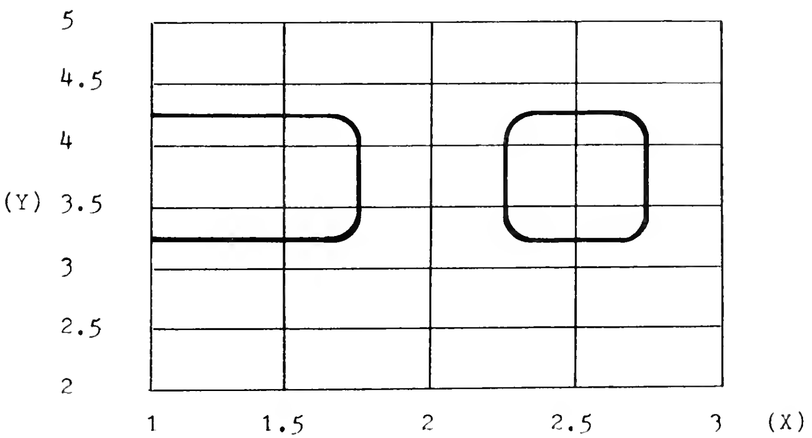


Fig. A.2.- Grid used in example. $f(x,y)=4.5$ locus' branches.

PTS NPTS = 15

1.	3.25
1.5	3.25
1.75	2.5
1.75	4.
1.5	4.25
1.	4.25
NA	NA
2.25	3.5
2.5	3.25
2.75	3.5
2.75	4.
2.5	4.25
2.25	4.
2.25	3.5
NA	NA

NDPTS

Fig. A.3.- Result variables for the locus in Fig. A.2

grid edge. Since in a NC x NR grid there are (NC-1)*NR + (NR-1)*NC edges and one 370 word holds 32 bits, thus the assertion about the dimension of IA in point 2 above.

Bits in IA are manipulated as follows: A bit is set to 1 by means of a call to the subroutine MARK whenever its corresponding grid edge is crossed by the locus to be built. Such circumstance is checked, for every edge but the extra ones, by the test

$$(A1 - ALEVL) * (A2 - ALEVL) \leq 0, \qquad (3)$$

where A1 and A2 are the function values at the edge extreme

---

(3) Refer to section 2 for test details in special cases.

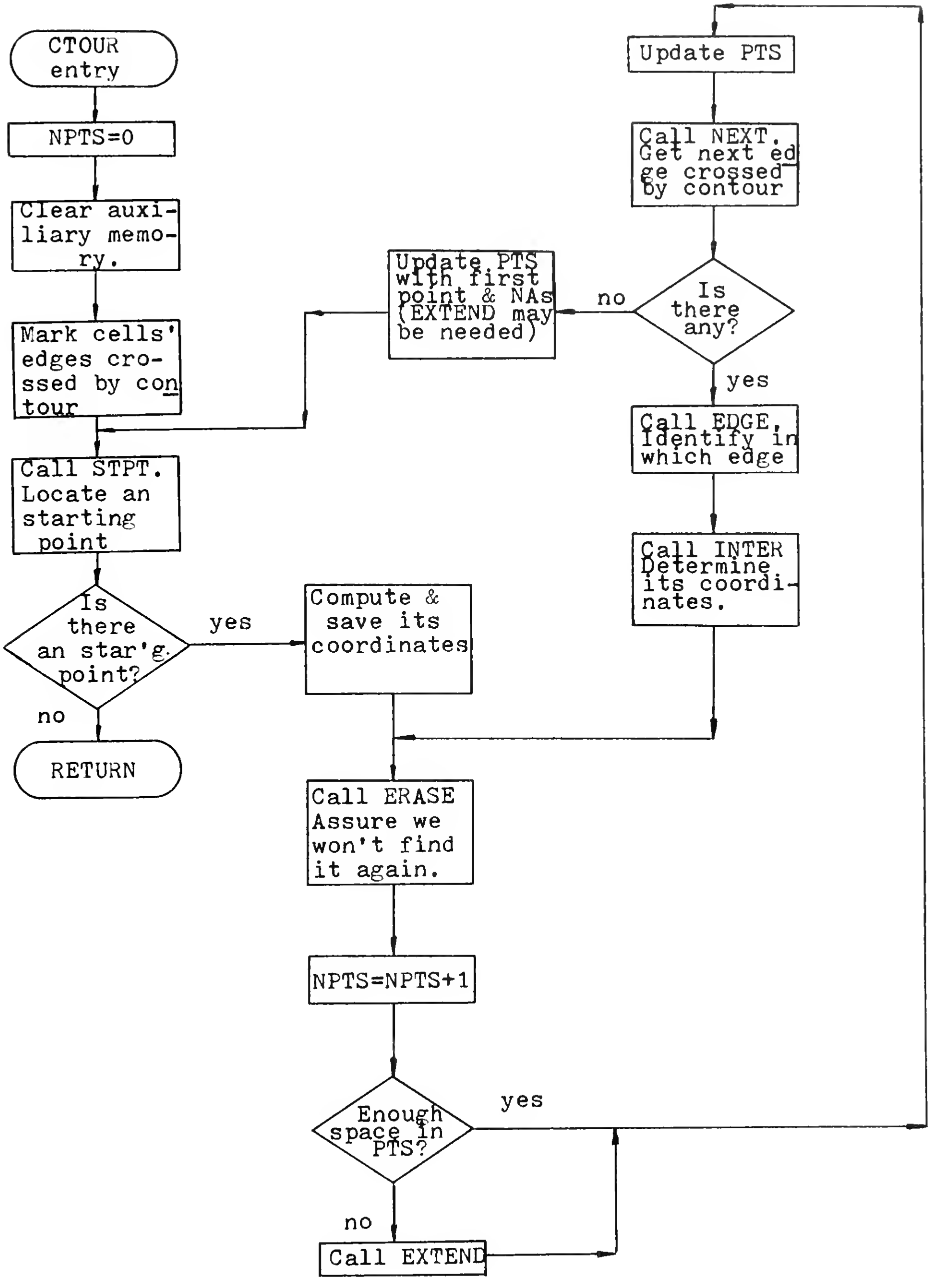


Fig. A.4.- CTOUR flowchart.

points.

The correspondence between IA bits and grid edges is established as follows: Edges are numbered as depicted in Fig. A.5 for the case of a 4 x 7 grid (vertical edges by rows first, then horizontal edges by rows). Having done this, given an edge number there is a corresponding bit in IA. For other uses, however,

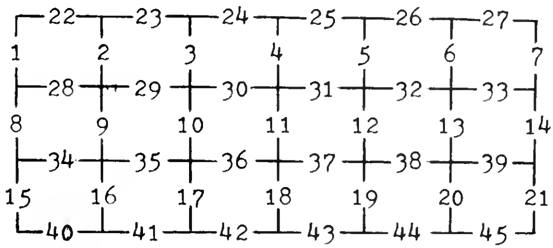


Fig. A.5.- Edge numbers in a 4 x 7 grid.

an edge is more conveniently identified as shown in Fig. A.6: The vertical edge joining the grid points (I,J) and (I+1,J) is speci-

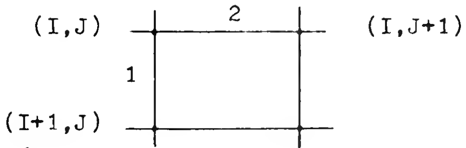


Fig. A.6.- A more convenient edge specification.

fied by means of the 3-tuple (I,J,1), and the horizontal one joining (I,J) to (I,J+1) by (I,J,2). The subroutine INDIA translates such specifications to edge numbers to manipulate the bit memory.

- The subroutine STPT scans the bit indicators in IA looking for 1's by means of calls to the subroutine SEE, which accepts a bit number and returns a variable $= \begin{Bmatrix} 1 \\ 2 \end{Bmatrix}$ if the bit is $\begin{Bmatrix} \text{a } 0 \\ \text{a } 1 \end{Bmatrix}$.

Boundary edges are checked first, to locate incomplete branches effectively. STPT returns an integer variable K equal to the number of a bit in IA currently set to 1. If no such a bit is found, K is returned set to zero.

- Calling the subroutine EDGE with a bit number K returns an edge specification in the form already seen in Fig. A.6, by means of a 3-tuple (I,J,KIN). (i.e., EDGE performs a translation in edge specification converse to that one done by INDIA).

- INTER interpolates linearly in the edge specified by (I,J,KIN) to end up with a point (XXX,YYY) such that f(XXX,YYY) = ALEVL (with the error proper of a linear interpolation; if such a method is considered ineffective for the function at hand, it is easy to write an alternative INTER subroutine to perform a finer interpolation, although this circumstance is unlikely to arise as other considerations -see section 2- force the grid cells to be of a size where interpolating linearly is usually enough.)

- ERASE accepts a bit number K in IA and sets it to zero.

- NEXT accepts an edge specified as (I,J,KIN) and looks at the adjacent cell searching for another edge crossed by the iso-contour under construction. If such an edge is not found, the original cell itself is checked for a crossed edge (because on the boundary it may happen that the adjacent cell does not have another crossed edge). When this search also fails, the contour branch being treated is complete.

Cells are identified by means of their north - west corner, and their edges by means of a variable set to 1, 2, 3 or 4 according to Fig. A.7. Notice that the 4 edges heve to be

identified here in order to effectively locate an adjacent cell.

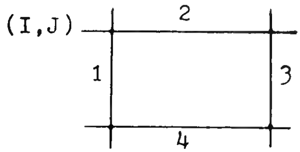


Fig. A.7.- Cell and edge identification for NEXT.

Given a cell as (I,J) and a value KIN(=1, 2, 3, 4) indicating which edge was crossed in that cell, NEXT identifies the adjacent cell as indicated in Fig. A.8 (where x indicates the cro-

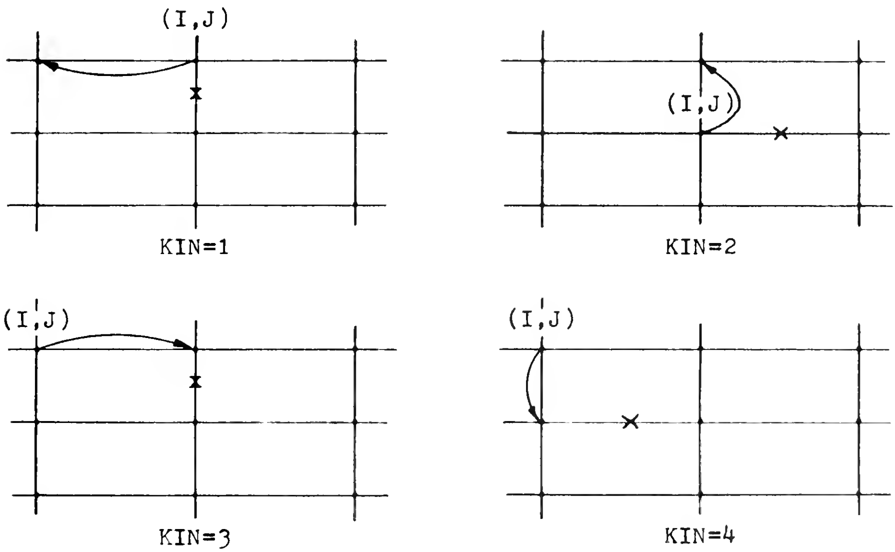


Fig. A.8.- Adjacent cell identification (NEXT).

ssed edge in cell (I,J) and the arrows point to the cell considered adjacent. In addition, NEXT updates I, J and KIN to specify the next edge to be analyzed; it also returns in K the bit number in IA corresponding to it. (K=0 indicates no more edges crossed by current locus' branch, thus specifying its end).

```
10
20
30
40
```

```
      SUBROUTINE CHECK(IA,I,J,KIN,NR,NC,K)
C****
C   ** CHECKS IF THE CELL (I,J) HAS A CROSSING EDGE OTHER THAN KIN,WHICH
C**** WAS ALREADY ERASED. IF SO, RETURNS IT AS (I,J,KIN) AND K. IF NOT,
C**** IT RETURNS K=0.
C****
      DIMENSION IA(1)
      KNEXT=KIN
      DO 30 L=1,3
      KNEXT=KNEXT+1
      IF(KNEXT-4) 20,20,10
   10 KNEXT=KNEXT-4
   20 K=INDIA(I,J,KNEXT,NR,NC)
      CALL SEE(IA,K,NOYES)
      GO TO(30,40),NOYES
   30 CONTINUE
      K=0
      RETURN
   40 KIN=KNEXT
      RETURN
      END
```

```
      SUBROUTINE CTOUR(A,NR,NC,XS,YS,ALEVL,IA,PTS,NDPTS,NPTS)
C    **
C**** RETURNS IN PTS ORDERED SETS OF POINTS BELONGIN. TO CONTOURS OF
C**** LEVEL ALEVL.
C**** ARGUMENTS.-
C****                A.-FUNCTION VALUES IN A GRID.-A(I,J) IS THE VALUE
C****                   F(XX,YY) WHERE XX=XS(J) & YY=YS(I).A MUST HAVE
C****                   NR ROWS & NC COLUMNS,AND THERE MUST BE 2 EXTRA
C****                   ROWS & 2 EXTRA COLUMNS(FIRST & LAST), FOR WHICH
C****                   IT IS NOT NECESSARY TO SPECIFY FUNCTION VALUES.
C****               NR.-SEE A.
C****               NC.-SEE A.
C****               XS.-SEE A.
C****               YS.-SEE A.
C****            ALEVL.-FUNCTION VALUE FOR WHICH A CONTOUR IS DESIRED.
C****               IA.-AUXILIARY MEMORY.ITS DIMENSION IN THE CALLING
C****                   PROGRAM SHOULD BE ((NC-1)*NR+(NR-1)*NC)/32+1
C****              PTS.-POINTS BELONGING TO THE GENERATED CONTOUR.
C****            NDPTS.-FIRST DIMENSION OF PTS IN THE CALLING PROGRAM.
C****                   (ITS SECOND IS 2).
C****             NPTS.-NUMBER OF ELEMENTS RETURNED IN PTS.
C****           EXTEND.-ROUTINE TO PROVIDE SPACE FOR PTS WHEN THE INITIAL
C****                   NDPTS SLOTS HAVE BEEN EXHAUSTED.
C****
      DIMENSION A(NR,NC),XS(NC),YS(NR),IA(1),PTS(NDPTS,2)
      DATA NA//FF0FFFFF/
      EQUIVALENCE (NA,ZNA)
      NPTS=0
C****
C**** CLEAR AUXILIARY MEMORY.
C****
      K=1+((NC-1)*NR+(NR-1)*NC)/32
      DO 10 I=1,K
   10 IA(I)=0
C****
C**** MARK CELLS' EDGES CROSSED BY CURRENT CONTOUR OF LEVEL ALEVL.
C**** FIRST AND LAST ROWS & COLUMNS ARE NOT MARKED(THEY ALLOW TO FINISH
C**** UP SUBCONTOURS CROSSING THE GRID BOUNDARIES).
C****
      NRM1=NR-1
      NCM1=NC-1
      DO 70 I=2,NRM1
      DO 70 J=2,NCM1
      IF(J-NCM1) 20,40,40
   20 IF((A(I,J)-ALEVL)*(A(I,J+1)-ALEVL)) 30,21,40
   21 IF(A(I,J)-ALEVL) 30,22,30
   22 IF(A(I,J+1)-ALEVL) 30,40,30
C****
```

```
C**** HORIZONTAL EDGE CROSSED.
C****
   30 K=INDIA(I,J,2,NR,NC)
      CALL MARK(IA,K)
   40 IF(I-NRM1) 50,70,70
   50 IF((A(I,J)-ALEVL)*(A(I+1,J)-ALEVL)) 60,51,70
   51 IF(A(I,J)-ALEVL) 60,52,60
   52 IF(A(I+1,J)-ALEVL) 60,70,60
C****
C**** VERTICAL EDGE CROSSED.
C****
   60 K=INDIA(I,J,1,NR,NC)
      CALL MARK(IA,K)
   70 CONTINUE
C****
C**** DETECT A CONTOUR STARTING POINT.
C****
   75 CALL STPT(IA,NR,NC,K)
      IF(K) 110,110,80
C****
C**** AN STARTING POINT WAS FOUND-IDENTIFY IN WHICH EDGE IS IT.
C****
   80 CALL EDGE(K,NR,NC,I,J,KIN)
C****
C**** DETERMINE AND SAVE THE COORDINATES OF THE STARTING POINT TO
C**** BE ABLE TO CLOSE IT WHEN ENDED.
C****
      CALL INTER(I,J,KIN,XS,YS,NR,NC,A,ALEVL,XXX,YYY)
      FIRSTX=XXX
      FIRSTY=YYY
      GO TO 86
C****
C**** IDENTIFY CROSSING POINT IN EDGE BY INTERPOLATION.
C****
   85 CALL INTER(I,J,KIN,XS,YS,NR,NC,A,ALEVL,XXX,YYY)
C****
C**** FORGET ABOUT THIS CROSSING POINT.
C****
   86 CALL ERASE(IA,K)
C****
C**** UPDATE POINTS SET(S).
C****
      NPTS=NPTS+1
      IF(MDPTS-NPTS) 87,90,90
   87 CALL EXTEND(PTS,MDPTS)
      NPTS=1
   90 PTS(NPTS,1)=XXX
      PTS(NPTS,2)=YYY
```

```
C****                                                              CT000970
C*** LOOK FOR NEXT CROSSING POINT.                                 CT000980
C****                                                              CT000990
      CALL NEXT(IA,NR,NC,I,J,KIN,K)                                CT001000
C****                                                              CT001010
C**** DOES IT EXIST? IF SO, IDENTIFY IT AND CONTINUE                CT001020
C****                                                              CT001030
      IF(K) 100,100,85                                             CT001040
C****                                                              CT001050
C**** IF NOT,TERMINATE A SUBTOUR & LOOK IF ANY OTHER EXISTS.        CT001060
C****                                                              CT001070
  100 IF(PTS(NPTS,1)-XS(2)) 101,109,101                            CT001080
  101 IF(PTS(NPTS,1)-XS(NCM1)) 102,109,102                         CT001090
  102 IF(PTS(NPTS,2)-YS(2)) 103,109,103                            CT001100
  103 IF(PTS(NPTS,2)-YS(NRM1)) 104,109,104                         CT001110
  104 NPTS=NPTS+1                                                  CT001120
      IF(NDPTS-NPTS) 105,106,106                                   CT001130
  105 CALL EXTEND(PTS,NDPTS)                                       CT001140
      NPTS=1                                                       CT001150
  106 PTS(NPTS,1)=FIRSTX                                           CT001160
      PTS(NPTS,2)=FIRSTY                                           CT001170
  109 NPTS=NPTS+1                                                  CT001180
      IF(NDPTS-NPTS) 107,108,108                                   CT001190
  107 CALL EXTEND(PTS,NDPTS)                                       CT001200
      NPTS=1                                                       CT001210
  108 PTS(NPTS,1)=ZNA                                              CT001220
      PTS(NPTS,2)=ZNA                                              CT001230
      GO TO 75                                                     CT001240
C****                                                              CT001250
C**** TERMINATE. RETURN TO CALLING PROGRAM.                         CT001260
C****                                                              CT001270
  110 RETURN                                                       CT001280
      END                                                          CT001290
```

```
      SUBROUTINE EDGE(K,NR,NC,I,J,KIN)
C****
C   ** TRANSLATES K-> I,J,KIN
C****
      KK=K-((NR-1)*NC)
      IF(KK) 10,10,20
   10 KIN=1
      I=((K-1)/NC)+1
      J=K-((I-1)*NC)
      RETURN
   20 I=((KK-1)/(NC-1))+1
      J=KK-((I-1)*(NC-1))
      KIN=2
      RETURN
      END
```

```
      FUNCTION INDIA(I,J,KIN,NR,NC)
C****
C   ** COMPUTES THE INDEX IN IA CORRESPONDING TO THE EDGE SPECIFIED
C**** BY I,J & KIN. NR & NC ARE NO. OF ROWS & COLUMNS OF A AS DECLARED
C**** IN THE CALLING PROGRAM(INCLUDING MANDATORY EXTRA ROWS AND COLS.)
C****
      II=I
      JJ=J
      KKIN=KIN
      GO TO(40,30,10,20),KKIN
   10 JJ=JJ+1
      KKIN=1
      GO TO 40
   20 II=II+1
      KKIN=2
   30 INDIA=(NR-1)*NC+(NC-1)*(II-1)+JJ
      RETURN
   40 INDIA=JJ+(II-1)*NC
      RETURN
      END
```

```
S
A
G
D
I
J
G
J
G
I
Y
D
P
X
X
R
X
D
P
Y
Y
R
F
```

```
      SUBROUTINE INTER(I,J,KIN,XS,YS,NR,NC,A,ALEVL,XXX,YYY)
C****
C   ** A CONTOUR POINT IS GENERATED,VIA LINEAR INTERPOLATION,LYING IN THE
C**** GRID EDGE DEFINED BY (I,J,KIN). THE RESULT IS (XXX,YYY).
C****
      DIMENSION XS(NC),YS(NR),A(NR,NC)
      II=I
      JJ=J
      GO TO(40,30,10,20),KIN
   10 JJ=JJ+1
      GO TO 40
   20 II=II+1
   30 YYY=YS(II)
      DELT=XS(JJ+1)-XS(JJ)
      P=(A(II,JJ+1)-A(II,JJ))/DELT
      XXX=(ALEVL-A(II,JJ))/P
      XXX=XXX+XS(JJ)
      RETURN
   40 XXX=XS(JJ)
      DELT=YS(II+1)-YS(II)
      P=(A(II+1,JJ)-A(II,JJ))/DELT
      YYY=(ALEVL-A(II,JJ))/P
      YYY=YYY+YS(II)
      RETURN
      END
```

```
STMT   SOURCE STATEMENT                                    ASM 0105 21.14 03/25/7

MARK     START                                                      MAR00001
         ENTRY FRASE,SEF                                            MAR00002
         STM   14,12,12(13)                                         MAR00003
         BALR  BASE,0                                               MAR00004
         USING *,BASE                                               MAR00005
AC       EQU   8                                                    MAR00006
ADDR     EQU   9                                                    MAR00007
BASE     EQU   10                                                   MAR00008
INDEX    EQU   11                                                   MAR00009
RONE     EQU   2                                                    MAR00010
MASK     EQU   3                                                    MAR00011
K        EQU   4                                                    MAR00012
         L     RONE,=F'1'                                           MAR00013
         LR    MASK,RONE                                            MAR00014
         SLL   MASK,31                                              MAR00015
         L     ADDR,4(0,1)       *GET K.                            MAR00016
         L     K,0(0,ADDR)       *                                  MAR00017
         LR    AC,K              .                                  MAR00018
         SR    AC,RONE           .                                  MAR00019
         SRL   AC,5              .GET CORRESPONDING ARRAY ELEMENT   MAR00020
         SLL   AC,2              .                                  MAR00021
         LR    INDEX,AC          .                                  MAR00022
         SLL   AC,3              *                                  MAR00023
         LR    6,AC              *                                  MAR00024
         LR    AC,K              *GET BIT POSITION                  MAR00025
         SR    AC,6              *                                  MAR00026
         SR    AC,RONE            .                                 MAR00027
         SRL   MASK,0(AC)         .PREPARE MASK                     MAR00028
         L     ADDR,0(0,1)        *                                 MAR00029
         L     AC,0(INDEX,ADDR)  *GET AFFECTED ARRAY WORD           MAR00030
         OR    AC,MASK            MARK SPECIFIED BIT                MAR00031
         ST    AC,0(INDEX,ADDR)   STORE WORD BACK                   MAR00032
         LM    14,12,12(13)      *RETURN                            MAR00033
         BR    14                *                                  MAR00034
FRASE    STM   14,12,12(13)                                         MAR00035
         BALR  BASE,0                                               MAR00036
         USING *,BASE                                               MAR00037
         L     RONE,=F'1'                                           MAR00038
         LR    MASK,RONE                                            MAR00039
         SLL   MASK,31                                              MAR00040
         L     ADDR,4(0,1)       *GET K                             MAR00041
         L     K,0(0,ADDR)       *                                  MAR00042
         LR    AC,K              .                                  MAR00043
         SR    AC,RONE           .                                  MAR00044
         SRL   AC,5              .GET CORRESPONDING ARRAY ELEMENT   MAR00045
         SLL   AC,2              .                                  MAR00046
         LR    INDEX,AC          .                                  MAR00047
         SLL   AC,3              *                                  MAR00048
         LR    6,AC              *                                  MAR00049
         LR    AC,K              *GET BIT POSITION                  MAR00050
         SR    AC,6              *                                  MAR00051
         SR    AC,RONE           .PREPARE MASK                      MAR00052
         SRL   MASK,0(AC)        .                                  MAR00053
         L     ADDR,0(0,1)        *GET AFFECTED ARRAY WORD          MAR00054
         L     AC,0(INDEX,ADDR)  *                                  MAR00055
```

```
    SOURCE STATEMENT                                                    ASM 0105 21.14 03/25/77

         XR    AC,MASK              ERASE SPECIFIED BIT                   MAR00056
         ST    AC,0(INDEX,ADDR)     STORE WORD BACK                       MAR00057
         LM    14,12,12(13)        *RETURN                                MAR00058
         BR    14                  *                                      MAR00059
SFF      STM   14,12,12(13)                                               MAR00060
         BALR  BASE,0                                                     MAR00061
         USING *,BASE                                                     MAR00062
         L     RONE,=F'1'                                                 MAR00063
         LR    MASK,RONE                                                  MAR00064
         SLL   MASK,31                                                    MAR00065
         L     ADDR,4(0,1)         *GET K                                 MAR00066
         L     K,0(0,ADDR)         *                                      MAR00067
         LR    AC,K                .                                      MAR00068
         SR    AC,RONE             .                                      MAR00069
         SRL   AC,5                .GET CORRESPONDING ARRAY ELEMENT       MAR00070
         SLL   AC,2                .                                      MAR00071
         LR    INDEX,AC            .                                      MAR00072
         SLL   AC,3                *                                      MAR00073
         LR    6,AC                *                                      MAR00074
         LR    AC,K                *GET BIT POSITION                      MAR00075
         SR    AC,6                *                                      MAR00076
         SR    AC,RONE             .PREPARE MASK                          MAR00077
         SRL   MASK,0(AC)          .                                      MAR00078
         L     ADDR,0(0,1)          *GET AFFECTED ARRAY WORD              MAR00079
         L     5,0(INDEX,ADDR)      *                                     MAR00080
         L     ADDR,8(0,1)          GET NOYES ADDRESS                     MAR00081
         NR    MASK,5               .CHECK WHETHER BIT WAS ON OR NOT      MAR00082
         BC    4,YES                .                                     MAR00083
         ST    RONE,0(0,ADDR)       *IF NOT,SET NOYES TO 1                MAR00084
         B     OUT                  *AND RETURN                           MAR00085
YES      SLL   RONE,1               .IF YES,SET NOYES TO 2                MAR00086
         ST    RONE,0(0,ADDR)       .                                     MAR00087
OUT      LM    14,12,12(13)         *RETURN                               MAR00088
         BR    14                   *                                     MAR00089
         END                                                              MAR00090
               =F'1'
```

```
SUBR

COMK
(I,J

DIME
II=I
JJ=J

ENGH

KKI(
IF(K
KKI(
GO T
JJ=J
GO
II=
GO
JJ=
GO
II=
CALL
IF(

WHE

CAL
RETU
I=I
J=J
KIN
RETU
END
```

```
      SUBROUTINE NEXT(IA,NR,NC,I,J,KIN,K)
C****
C*  * LOOKS FOR NEXT CROSSING EDGE, IF ANY & RETURNS IT IN BOTH FORMS
C**** (I,J,KIN) & K.-REFERRING TO IA-/RETURNS K=0 IF IT DOESN'T EXIST
C****
      DIMENSION IA(1)
      II=I
      JJ=J
C****
C**** EDGE OF ADJACENT CELL
C****
      KKIN=KIN+2
      IF(KKIN-4) 20,20,10
   10 KKIN=KKIN-4
   20 GO TO(30,40,50,60),KIN
   30 JJ=JJ-1
      GO TO 70
   40 II=II-1
      GO TO 70
   50 JJ=JJ+1
      GO TO 70
   60 II=II+1
   70 CALL CHECK(IA,II,JJ,KKIN,NR,NC,K)
      IF(K) 80,80,90
C****
C**** WHEN IT DOESN'T EXIST IN THE ADJACENT CELL,CHECK THE CURRENT ONE.
C****
   80 CALL CHECK(IA,I,J,KIN,NR,NC,K)
      RETURN
   90 I=II
      J=JJ
      KIN=KKIN
      RETURN
      END
```

```
. 7.
SUB
SCA
. RE
SI
AM
AM
NR
NC
DO
DO
IF
AM
GO
IF
AM
CO
J=
IF
AL
DO
AL
RE
EN
```

```
      SUBROUTINE SEARCH(A,NR,NC,ALEVL,NCNTR)
C****
C  ** SCANS THE TABLE A AND GENERATES NCNTR VALUES EQUALLY SPACED
C**** BETWEEN THE MAXIMUM AND MINIMUM VALUES OF A.
C****
      DIMENSION A(NR,NC),ALEVL(NCNTR)
      AMAX=A(1,1)
      AMIN=AMAX
      NRM1=NR-1
      NCM1=NC-1
      DO 40 I=2,NRM1
      DO 40 J=2,NCM1
      IF(A(I,J)-AMAX) 20,20,10
   10 AMAX=A(I,J)
      GO TO 40
   20 IF(A(I,J)-AMIN) 30,40,40
   30 AMIN=A(I,J)
   40 CONTINUE
      J=NCNTR-1
      DELTA=(AMAX-AMIN)/J
      ALEVL(1)=AMIN
      DO 50 I=1,J
   50 ALEVL(I+1)=ALEVL(I)+DELTA
      RETURN
      END
```

```
2.0

 SUBR

C
C LOOKS
C EDGES
C START
C
 DIMEN
 NRM1=
 NCM1=
 I=2
 DO 1
 K=INI
 CALL
 GO TO
 CONTI
 IF(I-
 I=NRM
 GO TO
 J=2
 DO 5
 K=INI
 CALL
 GO TO
 CONT
 IF(J-
 J=NC
 GO TO
 NRM2
 NCM2=
 DO 8
 DO 8
 DO 8
 K=INI
 CALL
 GO TO
 CONT
 K=0
 RETUR
 END
```

```
      SUBROUTINE STPT(IA,NR,NC,K)
C****
C  ** LOOKS FOR AN EDGE MARKED IN IA.IT GIVES PREFERENCE TO BOUNDARY
C**** EDGES,SO THAT INCOMPLETE SUBTOURS ARE LOCATED FIRST AND THEIR
C**** STARTING POINTS ARE ON THE BOUNDARIES.
C****
      DIMENSION IA(1)
      NRM1=NR-1
      NCM1=NC-1
      I=2
    5 DO 10 J=2,NCM1
      K=INDIA(I,J,2,NR,NC)
      CALL SEE(IA,K,NOYES)
      GO TO(10,90),NOYES
   10 CONTINUE
      IF(I-2) 20,20,30
   20 I=NRM1
      GO TO 5
   30 J=2
   40 DO 50 I=2,NRM1
      K=INDIA(I,J,1,NR,NC)
      CALL SEE(IA,K,NOYES)
      GO TO(50,90),NOYES
   50 CONTINUE
      IF(J-2) 60,60,70
   60 J=NCM1
      GO TO 40
   70 NRM2=NRM1-1
      NCM2=NCM1-1
      DO 80 I=2,NRM2
      DO 80 J=2,NCM2
      DO 80 L=1,2
      K=INDIA(I,J,L,NR,NC)
      CALL SEE(IA,K,NOYES)
      GO TO(80,90),NOYES
   80 CONTINUE
      K=0
   90 RETURN
      END
```

APPENDIX B

Routine invocation from TROLL. Macros.

The two TROLL macros listed below facilitate invocations to CTOUR from the TROLL system environment. (See TROLL manuals for details in specific commands*). To invoke these macros from any user account, it is necessary to specify the following command after logging in:

SEARCH MIT128_MACRO;

The macro &LIKPLOTS includes computation of likelihood function values at grid points, so that it is only useful for our particular problem. &PLOTS is more general, but it requires the function table to be available prior to its invocation.

Calling sequences examples are also included and are self-explanatory. (Capital letters correspond to system prompts).

---

*Available from the National Bureau of Economic Research, 545 Technology Sq., Cambridge.

```
&ERROR &GOTO ERR
SEARCH HILL_FUNCTION:SEARCH WILK_FUNCTION:
OUTOPT DEVICE T4010;
OUTOPT NOMARKS:
OUTOPT &0"'GRID' OR 'NOGRID':":
&READ &4"YOUR MU VALUES:" &END
&READ &5"YOUR SIGMA SQ. VALUES:" &END
DO R&4=EXPAND(&4,NOB(&5),NOB(&5)):DO R&4=TRANSP(R&4):
DO R&5=EXPAND(&5,NOB(&4),NOB(&4)):
&READ &7"B VALUES:" &END
&READ &8"OBSERVATIONS:" &END
&READ &9"N VALUE:" &END
DO &8&9=LIKLHOOD(&7,R&4,R&5,COMBINE(&9),&8):
DO IFARG(1)=MAXS(&8&9);
&PRINTA
YOUR MAXIMUM VALUE  IS &IFARG(1).
CHOOSE YOUR CONTOUR LEVELS ACCORDINGLY.
&END
&READ &2"NEW LEVELS?" &END
&IF &2 CEQ NO &GOTO OLD &IFEND
DELETE DATA &3"NEW LEVELS NAME:":
&SIGA:
DEDIT &3,1,1:
ADD TOP,&0"ENTER LEVELS IN QUOTES:":
FILE:
&GOTO NEXT
&OLD:
&READ &3"OLD LEVELS NAME:" &END
&NEXT:
DO &3=REV(SORT(&3,&3));
DO SETIFARG(&3,50):
DO &8&9C=NEWCIRLE(&4,&5,&8&9,&3);
DO &4.=&8&9C_X:
DO &5.=&8&9C_Y:
DO IFARG(2)=NOB(&8&9C);
DO SETIFARG(&8&9C,5);
&SET &IFARG(3)=1 &END
&SET &IFARG(4)=5 &END
&SETC &CIFARG(1)=ABCDEFGHIJKLMNOPORSTUVWXYZ &END
CLOODS:
&SET &IFARG(1)=1 &END
DEFSPACE &4. &5.:
&READ &0"VARIABLE IN HORIZONTAL AXIS:" &END
&IF &0 CEQ "&5" &GOTO REVES &IFEND
PPLANE 1 2:
&GOTO DIBUIX
&REVES:
PPLANE 2 1:
&DIBUIX:
&SETC &CIFARG(2)=&KEEPL 1 &CIFARG(1) &END
&SETC &CIFARG(1)=&STRIPL 1 &CIFARG(1) &END
MARK &IFARG(1) &CIFARG(2):
&SET &IFARG(1)=&IFARG(&IFARG(4))+1 &END
&IF &IFARG(3) LT &IFARG(2)
&SET &IFARG(3)=&IFARG(3)+1 &END
&SET &IFARG(4)=&IFARG(4)+1 &END
&GOTO DIBUIX
&IFEND
SCAL:
&PRINTA
/&3(9/FIRST 3 CONTOUR VALUES ARE(THE REST AT THE FILE  3):
```

```
A=&TFARG(50).B=&TFARG(51).C=&TFARG(52).
&END
&EXIT
&ERR:
&IF &TFARG(0) NE 13018 &ABORT &TFLOD
&GOTO STGA
```

PLOTS -

```
&ERROR &GOTO ERR
SEARCH FIRST RICE_SEGMENT:
SEARCH UTIL_FUNCTION:
OUTPUT DEVICE 14010:
OUTPUT NOMARKS:
OUTPUT &0"'GRID' OR 'NOGRID':":
DO TEARG(1)=MAXS(&1"YOUR TABLE:");
&PRINTA
THE MAXIMUM VALUE IN &1 IS &TEARG(1).
CHOOSE YOUR CONTOUR LEVELS ACCORDINGLY.
&END
&READ &2"NEW LEVELS?" &END
&IF &2 CEQ NO &GOTO OLD &IFEND
DELETE DATA &3"NEW LEVELS NAME:";
&SIGA:
UPDATE &3.1.1:
ADD TOP.&0"ENTER LEVELS IN QUOTES:";
FILE:
&GOTO NEXT
&OLD:
&READ &3"OLD LEVELS NAME:" &END
&NEXT:
DO &3=REV(SORT(&3,&3)):
DO SETTEARG(&3.50)):
&READ &4"YOUR X VALUES:" &END
&READ &5"YOUR Y VALUES:" &END
&READ &6"YOUR CONTOUR NAME:" &END
DO &6=NEWCURVE(&4,&5,&1.&3):
DO &4.=&6_X:
DO &5.=&6_Y:
DO TEARG(2)=NUM(&7):
DO SETTEARG(&6.5):
&SET &TEARG(3)=1 &END
&SET &TEARG(4)=5 &END
&SETC &CTEARG(1)=ABCDEFGHIJKLMNOPQRSTUVWXYZ &END
CLOUDS:
&SET &TEARG(1)=1 &END
DEFSPACE &4. &5.:
&READ &0"VARIABLE ON HORIZONTAL AXIS:" &END
&IF &0 CEQ "&5" &GOTO REVES &IFEND
PPLANE 1 2:
&GOTO UPDATX
&REVES:
PPLANE 2 1:
UPDATX:
&SETC &CTEARG(2)=&KEEPL 1 &CTEARG(1) &END
&SETC &CTEARG(1)=&STRIPL 1 &CTEARG(1) &END
MARK &TEARG(1) &CTEARG(2):
&SET &TEARG(1)=&TEARG(&TEARG(4))+1 &END
&IF &TEARG(3) LT &TEARG(2)
&SET &TEARG(3)=&TEARG(3)+1 &END
&SET &TEARG(4)=&TEARG(4)+1 &END
&GOTO UPDATX
&IFEND
SCAT:
&PRINTA
/&1/ THE FIRST 3 CONTOUR VALUES ARE(THE REST AS IN FILE &3):
A=&TEARG(50).B=&TEARG(51).C=&TEARG(52).
&END
TOUT:
```

```
TROLL COMMAND: do mu = seq(0.,3.,0.1);

TROLL COMMAND: do sigma = seq(0.01,3.,0.1);

TROLL COMMAND: &likplots


'GRID' OR 'NOGRID':grid

YOUR MU VALUES:mu

YOUR SIGMA SQ. VALUES:sigma

B VALUES:bb

OBSERVATIONS:nsob

N VALUE;300

%YOUR MAXIMUM VALUE  IS -128.125.
CHOOSE YOUR CONTOUR LEVELS ACCORDINGLY.
NEW LEVELS?yes

NEW LEVELS NAME:lev300

%NEW SERIES LEV300
ENTER LEVELS IN QUOTES:"-128.125 -128.5 -128.8 -129 -130 -135"

VARIABLE IN HORIZONTAL AXIS:sigma
```

```
TROLL COMMAND: &plots

'GRID' OR 'NOGRID':grid

YOUR TABLE:nsob300

THE MAXIMUM VALUE IN NSOB300 IS -128.125.
CHOOSE YOUR CONTOUR LEVELS ACCORDINGLY.
NEW LEVELS?no

OLD LEVELS NAME:lev300

YOUR X VALUES:mu

YOUR Y VALUES:sigma

YOUR CONTOUR NAME:nsob300c

%VARIABLE IN HORIZONTAL AXIS:sigma
```

APPENDIX C

Program requirements and performance.

The memory requirements for the subroutines included in the iso - contour generating package are as follows:

Subroutine	Bytes
CHECK	684
CTOUR	2,552
EDGE	604
INDIA	628
INTER	1,134
MARK, ERASE, SEE	240
NEXT	822
SEARCH	896
STPT	1,088
Total	8,648

As for execution times, we include below the CPU times taken to generate and plot iso - contours in different cases. We measured them for different grid sizes (given by the number of grid nodes) and different number of contours to generate. Actually, these times are execution times for the macro &PLOTS, and thus they include some TROLL overhead. They are measured in seconds.

Grid Size	Contours generated and plotted	
	3	6
25 x 25	0.67	0.92
50 x 50	1.42	2.34
75 x 75	2.43	4.28
100 x 100	3.82	6.91

References.

(1) Eytan Barouch and Gordon M. Kaufman, "Sampling Without Replacement And Proportional To Size", September, 1974

(2) G. Cottafava and G. Le Moli, "Automatic Contour Map", Communications of the ACM, July, 1969.

(3) C. M. Crame, "Contour Plotting For Functions Specified At Nodal Points of an Irregular Mesh Based on an Arbitrary Two Parameter Co-ordinate System", The Computer Journal-Algorithms Supplement,

(4) E.M. Greenwalt, "Contours of a Function of Two Variables", University of Texas, 1968.